UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA SUPERIOR DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

Diseño e Implementación de un Sistema de Mensajería Multimedia para dispositivos Nokia N800.

AUTOR: MARÍA JESÚS MARTÍNEZ VEGA

TUTOR: ANTONIO DE LA OLIVA DELGADO

Febrero 2010

A Antonio, mi tutor, por toda la ayuda que me ha proporcionado durante el desarrollo del proyecto y su paciencia.

A Omar y Jose Luis, que me enseñaron a discernir lo absurdo de lo valiente y a que vivir sin emocionar o sin emocionarse no tiene ningún sentido.

A Juancho y Jose, por su cariño, comprensión y fe, por enseñarme el valor de la amistad y porque ya forman parte de mi misma y de mi credo.

A mis padres y a mi hermana. Por su apoyo incondicional y su cariño. Porque sin ellos no sería lo que soy. Porque me siento orgullosa de quiénes son y de cómo son.

A Aurelio, porque cambió mi vida y me cambió para siempre. Por su apoyo constante y por tantas otras cosas que han hecho que sea feliz.

“Hay una gran diferencia entre conocer el camino y andar el camino.”

Morpheo. Matrix.

NO TE DETENGAS

No dejes que termine el día sin haber crecido un poco,
sin haber sido feliz, sin haber aumentado tus sueños.
No te dejes vencer por el desaliento.
No permitas que nadie te quite el derecho a expresarte,
que es casi un deber.
No abandones las ansias de hacer de tu vida algo extraordinario.
No dejes de creer que las palabras y las poesías
sí pueden cambiar el mundo.
Pase lo que pase nuestra esencia está intacta.
Somos seres llenos de pasión.
La vida es desierto y oasis.
Nos derriba, nos lastima,
nos enseña,
nos convierte en protagonistas
de nuestra propia historia.
Aunque el viento sople en contra,
la poderosa obra continúa:
Tu puedes aportar una estrofa.
No dejes nunca de soñar,
porque en sueños es libre el hombre.
No caigas en el peor de los errores:
el silencio.
La mayoría vive en un silencio espantoso.
No te resignes.
Huye.
"Emito mis alaridos por los techos de este mundo",
dice el poeta.
Valora la belleza de las cosas simples.
Se puede hacer bella poesía sobre pequeñas cosas,
pero no podemos remar en contra de nosotros mismos.
Eso transforma la vida en un infierno.
Disfruta del pánico que te provoca
tener la vida por delante.
Vívela intensamente,
sin mediocridad.
Piensa que en ti está el futuro
y encara la tarea con orgullo y sin miedo.
Aprende de quienes puedan enseñarte.
Las experiencias de quienes nos precedieron
de nuestros "poetas muertos",
te ayudan a caminar por la vida
La sociedad de hoy somos nosotros:
Los "poetas vivos".
No permitas que la vida te pase a ti sin que la vivas ...

Walt Whitman (1819-1892)

Resumen

Los SMS aparecieron, básicamente, como un servicio de notificación por parte de la operadora de nuevas llamadas en el buzón de voz, de las tarifas de roaming en un nuevo país, etc. Pero en los últimos años, el SMS se ha convertido en un gran negocio, debido especialmente a su utilización masiva por parte de los adolescentes, que han hecho de los mensajes cortos un código de comunicación entre ellos que ha sorprendido al sector, pues ni ha sido un servicio cómodo de usar ni ha habido una estrategia clara de marketing por parte de las operadoras.

Después del éxito arrollador del servicio SMS, la tecnología MMS se perfila como su continuación natural, ya que añade a las ventajas de los mensajes cortos, contenidos multimedia tales como imágenes y sonidos. De hecho, los proveedores de contenidos encuentran un mayor potencial en MMS que en SMS, donde el 90% de los mensajes son de persona a persona.

En este contexto, no cabe duda que las aplicaciones MMS presentan un gran potencial de negocio por lo que el conocimiento de los protocolos a nivel docente y la práctica en aplicaciones basadas en este sistema de comunicación, representan un acierto para el alumnado o potencial equipo de desarrolladores en el sector de las telecomunicaciones.

Índice general

Lista de figuras

Lista de tablas

1. INTRODUCCIÓN

El **departamento de Ingeniería Telemática** de la **Universidad Carlos III de Madrid** [7] está formado por un equipo consolidado de Doctores e Ingenieros de Telecomunicación e Informática de cerca de 100 personas. Este equipo cuenta con un amplio historial de actividad, reconocimiento y experiencia contrastada a nivel nacional e internacional en proyectos de investigación y desarrollo, formación y consultoría. Estas actividades se realizan para, o en colaboración con, empresas de servicios, fabricantes y administraciones públicas.

Dentro de su amplia labor docente, cuya información íntegra puede obtenerse en la web oficial del departamento http://www.it.uc3m.es , se enmarca la signatura de Laboratorio de Ingeniería de Servicios, continuación natural de la asignatura Ingeniería de Servicios. Esta asignatura, de naturaleza eminentemente práctica, fomenta el refuerzo del conocimiento, adquirido en su predecesora, mediante la realización de prácticas de laboratorio.

Su plan formativo o programa (accesible en http://www.it.uc3m.es/~isoto/asignaturas/lis) incluye, entre otros, el desarrollo de aplicaciones en entornos de redes de comunicaciones celulares (WAP, Android, MMS). El proyecto nace de la necesidad de una herramienta formativa técnicamente motivadora para el alumnado que ofrezca el soporte práctico para el desarrollo del punto anteriormente citado.

1.1. Objetivos

Partiendo de las necesidades identificadas dentro del marco de proyecto para el departamento de ingeniería telemática de la Universidad Carlos III y fruto de las reuniones mantenidas con su representante, los principales requisitos a satisfacer por el sistema propuesto pueden resumirse en:

Desarrollo de una aplicación capaz de enviar y recibir mensajes multimedia (MMS) en la plataforma ofrecida por los terminales Nokia N800 Internet Table. Los requisitos específicos se desgranarán en el tercer capítulo dedicado al análisis funcional de la aplicación.

Para lograr los objetivos propuestos, se aplicará un ciclo de vida en cascada, con las siguientes fases:

1. Obtención de nuevos requisitos mediante reuniones con el cliente, en este caso, la Universidad Carlos III de Madrid. Como resultado elaboraremos el análisis funcional.

2. Definición del modelo arquitectónico de la aplicación y evaluación de las tecnologías a utilizar. Como resultado elaboraremos el análisis Técnico.

3. Desarrollo de la aplicación. Como resultado elaboraremos el manual de desarrollo y el manual de usuario.

4. Despliegue y puesta en producción. Como resultado elaboraremos los informes de pruebas integradas y el documento de implantación en producción.

1.2. Estructura del proyecto

En este capítulo se ha ofrecido una breve introducción al proyecto presentándose las motivaciones de su puesta en marcha, así como una breve descripción del sistema a desarrollar y la metodología a emplear en el desarrollo del mismo.

En el capítulo dos, **Estado del arte**, se abordará el estado del arte, es decir, se describirá brevemente la situación actual en el campo del desarrollo de aplicaciones para dispositivos móviles y las nuevas tecnologías existentes.

En el capítulo tres, **Análisis funcional**, se afrontará la especificación y análisis de la aplicación así como el diseño de la interfaz.

En el capítulo cuatro, **Análisis Técnico**, se describirán con cierto detalle las tecnologías, frameworks[1], entornos de desarrollo y lenguaje de programación utilizados

[1] ***Framework***: Colección de clases que proporciona un conjunto de servicios para un dominio particular, exportando un cierto número de clases y mecanismos de utilización que pueden utilizarse tal cual o adaptarse para una aplicación concreta.

para la implementación de la aplicación. De igual manera, se presentarán aspectos relevantes dentro de la fase de diseño como la arquitectura seleccionada.

En el capítulo quinto, **Pruebas**, se describirán las pruebas unitarias así como las pruebas integradas y UAT (User Acceptance Testing o beta testing).

En el capítulo sexto, **Entregables**, se presentará el manual de usuario.

En el capítulo séptimo, **Gestión del proyecto**, se describirán la planificación y costes. También se describirá el plan de puesta en producción que incluye las tareas para la implantación de la aplicación. Así mismo se describirán los problemas encontrados durante la implantación y las soluciones tomadas para la resolución de los mismos.

El capítulo octavo y último, **Conclusiones**, se dedicará a las conclusiones del proyecto y a posibles recomendaciones futuras para una nueva versión de esta aplicación. Así mismo, se realizará una valoración final del proceso de desarrollo desde un punto de vista personal.

Por último, se han incluido las referencias bibliográficas, el glosario y los apéndices.

2. ESTADO DEL ARTE

2.1. Desarrollo de aplicaciones en dispositivos móviles

2.1.1. INTRODUCCIÓN

Si pensamos en algún dispositivo móvil [6], lo primero en lo que pensamos es en un teléfono móvil, y como mucho en una PDA. Pero en la actualidad son varios los dispositivos móviles disponibles en el mercado. Este es el caso de los PC's portátiles, de los teléfonos con acceso a Internet, PocketPC's, NetPC's, etc.

Este hecho da lugar a una importante problemática para quien programa tales dispositivos, ya que cada uno de ellos tienen unas características particulares: cada uno dispone de una memoria determinada o ha de soportar un lenguaje y un entorno específicos.

El problema de los programadores de estos dispositivos reside, principalmente, en el momento en el que quieren estandarizar todos los dispositivos. Esto es muy difícil, ya que cada uno de ellos tiene unas aplicaciones diferentes, está implementado en un lenguaje distinto y utiliza técnicas de desarrollo y sistemas operativos dispares.

2.1.2. GENERALIDADES DE LOS DISPOSITIVOS MÓVILES

Un dispositivo móvil (DM) [6], se puede definir como aquel que disfruta de autonomía de movimiento y está libre de cableado. La principal característica de un dispositivo móvil es su gran capacidad de comunicación, la cual permite tener acceso a información y servicios independientemente del lugar y el momento en el que nos encontremos. Es decir, es una fuente de información fácil de transportar.

La movilidad de un DM está condicionada por la necesidad de utilizar una batería. Esto representa un inconveniente debido a que la batería necesita recargas periódicas, lo que dificulta en muchos casos la portabilidad del DM.

Un dispositivo móvil se caracteriza, en general, por su reducido tamaño, el cual aporta una ventaja notable: favorece la movilidad de los DM's. A su vez, comporta una serie de inconvenientes, como son que han de utilizar un procesador más simple y una memoria pequeña. Además, la interfaz con el usuario también es reducida, ya que la

mayor parte de los DM's tienen una pantalla reducida, un teclado muy pequeño, o carecen de él, reconocimiento de voz limitado, etc.

2.1.3. TIPOS DE APLICACIONES MÓVILES

2.1.3.1. Aplicaciones Web Móviles

Una aplicación web móvil, es una aplicación igual que la que podemos ejecutar en un PC de escritorio pero en el mundo móvil, con una versión más reducida del lenguaje.

Las características de las aplicaciones [6] son:

- Accesibilidad a través de un navegador web, parecido a un navegador de PC de escritorio, pero con un código mucho más limitado.
- Las aplicaciones han de ser interactivas y dinámicas, ya que muchos móviles tienen el inconveniente de tener un teclado muy limitado y una pantalla reducida.
- La interfaz con el usuario ha de ser rica.
- El modelo de propagación por la red es el mismo que se utiliza para la web pero a través de la interfaz móvil.
- El móvil sólo se encarga de la interfaz gráfica con el usuario. Esto provoca que el dispositivo tenga capacidad innata de comunicación y siempre esté on-line para poder ejecutar una aplicación dinámica.
- La lógica de procesado pesado se realiza en el servidor y el resultado se manda al móvil que sólo se encarga de la interfaz gráfica.

Este tipo de aplicaciones se pueden crear tanto con Java como con .NET, como veremos en las secciones dedicadas a estas plataformas de desarrollo en el apartado 2.4.

2.1.3.2. Aplicaciones Standalone

Aplicación que, al contrario que la aplicación web móvil, se instala y ejecuta directamente en el dispositivo.

Esta aplicación comporta la ventaja de que el dispositivo no tiene por qué tener una capacidad nata de comunicación y puede estar completamente desconectado de la red si la aplicación no requiere ningún tipo de acceso al servidor (off-line). De otra

manera, si es necesario para el desarrollo de la aplicación, el dispositivo puede estar conectado (on-line).

También, y como complemento a lo anterior, si la aplicación se va a ejecutar completamente en el dispositivo, éste ha de tener mucha más memoria y más potencia de cálculo de CPU para poder llevar a cabo todo el cómputo, ya que se necesitará más recursos. Por otra parte, el dispositivo cliente ha de decidir en todo momento cómo va a desarrollarse la aplicación, por lo que el cliente ha de ser inteligente (denominado Smart Client) y la persona que posee el terminal ha de tener mucho más control sobre él (se ha de tener un conocimiento amplio sobre el manejo del terminal).

Por último, sobre este tipo de aplicaciones, se ha de comentar que el modelo de propagación de datos es similar al de un ordenador de escritorio pero mediante una red inalámbrica.

2.1.3.3. Aplicaciones de Notificación

Aplicaciones basadas en la comunicación [6], que consisten en el envío y recepción de mensajes.

Estos mensajes pueden ser noticias, conversaciones de chat, o información temática solicitada en un determinado lugar. Los tipos pueden ser:

• SMS: mensaje escrito con el teléfono móvil.

• EMS: mensaje que permiten introducir gráficos.

• MMS: mensaje que permite introducir una aplicación multimedia.

Por otro lado, este tipo de aplicaciones tiene una interfaz con el usuario muy simple, capaz de funcionar en casi cualquier tipo de dispositivo por simple que sea. Para que el dispositivo pueda funcionar ha de estar siempre conectado a la red (on-line) y es imprescindible que tenga capacidad innata de comunicación.

La aplicación que se ha desarrollado como consecuencia de los requerimientos del departamento de Ingeniería Telemática para la asignatura de LIS y sobre la que hablaremos en el presente documento podría verse como una mezcla de aplicación Standalone y de aplicación de Notificación como veremos más adelante.

2.2. Descripción de servicios de mensajería

2.2.1. SERVICIO DE MENSAJE CORTO (SHORT MESSAGE SERVICE)

El servicio de mensajes cortos o SMS, del inglés Short Message Service, es un servicio disponible en los teléfonos móviles para el envío de mensajes de texto entre celulares, teléfonos fijos y otros dispositivos de mano.

SMS fue diseñado originariamente como parte del estándar GSM (Groupe Spécial Mobile o Sistema Global para las Comunicaciones Móviles), nacido en Europa, pero luego se extendió a las redes TDMA (Time Division Multiple Access) y CDMA (Code Division Multiple Access).

En el estándar **GSM**, los SMS se transmitían a través del canal de señalización de la red y su finalidad era, en un principio, servir de medio para que los operadores de red enviaran información sobre el servicio a los abonados (llamadas perdidas, mensaje en el buzón de voz, etc…), sin que éstos pudieran responder ni enviar mensajes a otros clientes. Este tipo de mensajes se denominaban **MT-SM** (Mobile Terminated-Short Message, es decir, mensajes que llegan al terminal del usuario). Sin embargo, la empresa Nokia desarrolló un sistema para permitir la comunicación bidireccional por SMS. Los mensajes enviados por los usuarios pasaron a denominarse **MO-SM** (Mobile Originated, originados en el terminal del usuario).

La arquitectura GSM [12] está constituida por tres partes:

1. **Subsistema Radio** (**RSS**, **R**adio **S**ub**S**ystem). Cubre la comunicación entre las estaciones móviles (**MS**) y las estaciones base (**BS**).
2. **El subsistema de estaciones base** (**BSS**), incluido dentro de la parte Radio, está constituido por los siguientes elementos:
    a. **BTS** (**B**ase **T**ransceiver **S**tation): emisor, receptor y antena.
    b. **BSC** (**B**ase **S**tation **C**ontroller): Handover, control de las **BTS**, mapeo de canales radio sobre los canales terrestres.
3. **Subsistema de red y conmutación** (**NSS**, **N**etwork and **S**witching **S**ubsystem). Conmutación, gestión de la movilidad, interconexión con

otras redes y control del sistema. Esta es la parte más compleja, siendo sus elementos fundamentales los siguientes:

a. **MSC** (**M**obile Services **S**witching **C**enter), centro de conmutación entre otras muchas funciones.

b. **GMSC** (**G**ateway **M**obile Services **S**witching **C**enter). Conexión con otras redes.

c. **Bases de datos**:

   i. **HLR** (**H**ome **L**ocation **R**egister)

   ii. **VLR** (**V**isitor **L**ocation **R**egister)

   iii. **EIR** (**E**quipment **I**dentity **R**egister)

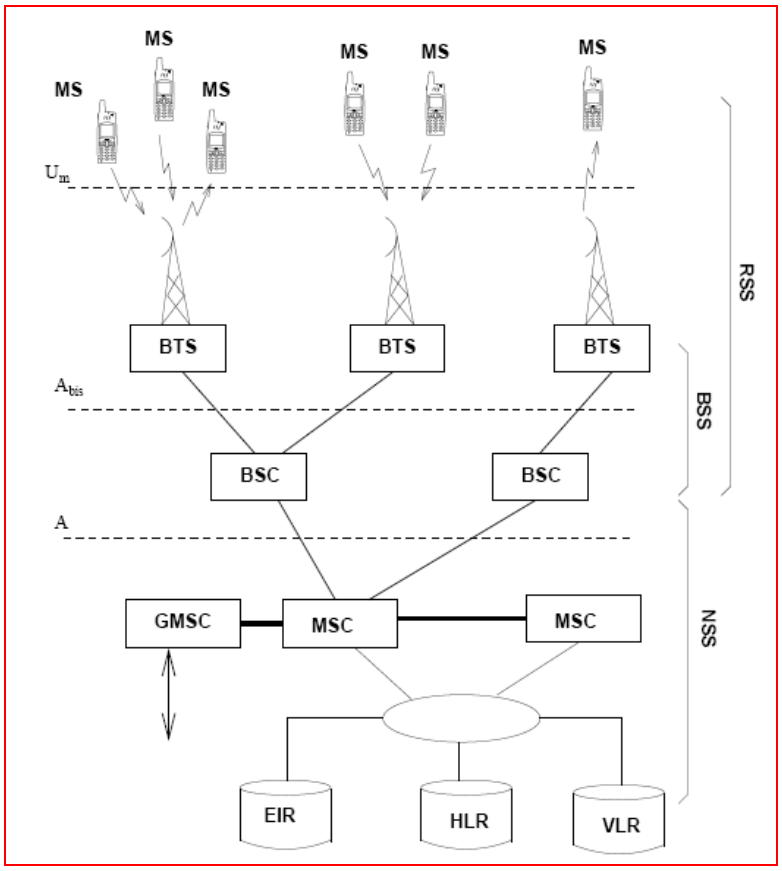


Ilustración 1 - Arquitectura red GSM [12]

La estructura de un mensaje SMS, denominada *unidad de datos de protocolo* (PDU), se clasifica en seis tipos según el sentido de transferencia del mensaje, teniendo un formato distinto para cada uno de ellos:

- **SMS-DELIVER**: Transmitir un mensaje desde el **SC** al **MS**
- **SMS-DELIVER-REPORT**: Error en la entrega (si lo ha habido)
- **SMS-SUBMIT**: Trasmitir un mensaje corto desde el **MS** al **SC**
- **SMS-SUBMIT-REPORT**: Error en la transmisión (Si lo ha habido)
- **SMS-STATUS-REPORT**: Transmitir un informe de estado desde el **SC** al **MS**
- **SMS-COMMAND**: Transmitir un comando desde el **MS** al **SC**

El mostrado en la ilustración 2 corresponde a SMS-SUBMIT, en el cuerpo caben hasta 153 (160 – 7) caracteres por mensaje codificado en GSM de 7 bits, pero pueden concatenarse varias páginas hasta un máximo de 39015(255*153) caracteres en un mensaje no comprimido. El estándar de SMS establece un sistema de compresión opcional que permitiría enviar mensajes de más de 160 caracteres.

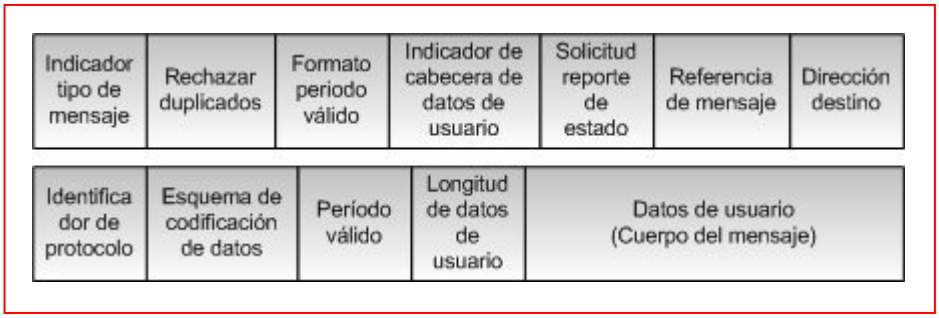


Ilustración 2- Estructura de un mensaje SMS (PDU) [8].

Como podemos observar en la ilustración, cada mensaje SMS incluye, además del texto que se ha escrito, la fecha de envío, los números de teléfono del remitente y el destinatario, así como el del centro (SMSC) desde el que se envía, entre otros.

El primer SMS comercial se envió el 3 de diciembre de 1992 en Reino Unido, entre Neil Papworth y Richard Jarvis, técnicos de la red Vodafone. El texto era “Merry Christmas”. Pero el servicio SMS tardó unos años en popularizarse debido, sobre todo, a los fraudes, ya que los usuarios modificaban el número del SMSC a través del que enviaban los mensajes por el de una operadora distinta que ofrecía SMS gratuitos, evitando así pagar los mensajes enviados.

Son muchos los fraudes a los que ha tenido que enfrentarse esta tecnología. Uno de los más utilizados fue una práctica llamada SMS Spoofing, en el que se engaña a la red para enviar mensajes sin ningún coste. Consiste en suplantar la identidad de un extranjero para enviar SMS al país, supuestamente de origen. El "spoofing" genera

problemas de facturaciones equivocadas y bloqueos de la red. Mediante esta técnica se llevaron a cabo ataques a nivel internacional.

A grandes rasgos, la red SMS, de la cual podemos ver una esquematización en la ilustración 3, está compuesta de los siguientes elementos:

- La *torre celular* o *estación base* (BS), aquella con la que se comunica directamente el dispositivo móvil.
- El *centro de conmutación móvil* (MSC), que coordina y controla la configuración de la llamada y el direccionamiento entre el celular y el área de servicio.
- El *centro SMS* (SMSC), que es el encargado de almacenar los mensajes y reenviarlos cuando el destinatario esté disponible, garantizando así la entrega.
- El SMSC, que se comunica con las redes TCP/IP a través de la *compuerta MSC* (GMSC), obtiene la información de direccionamiento y entrega el mensaje al MSC visitado de la unidad móvil de destino.

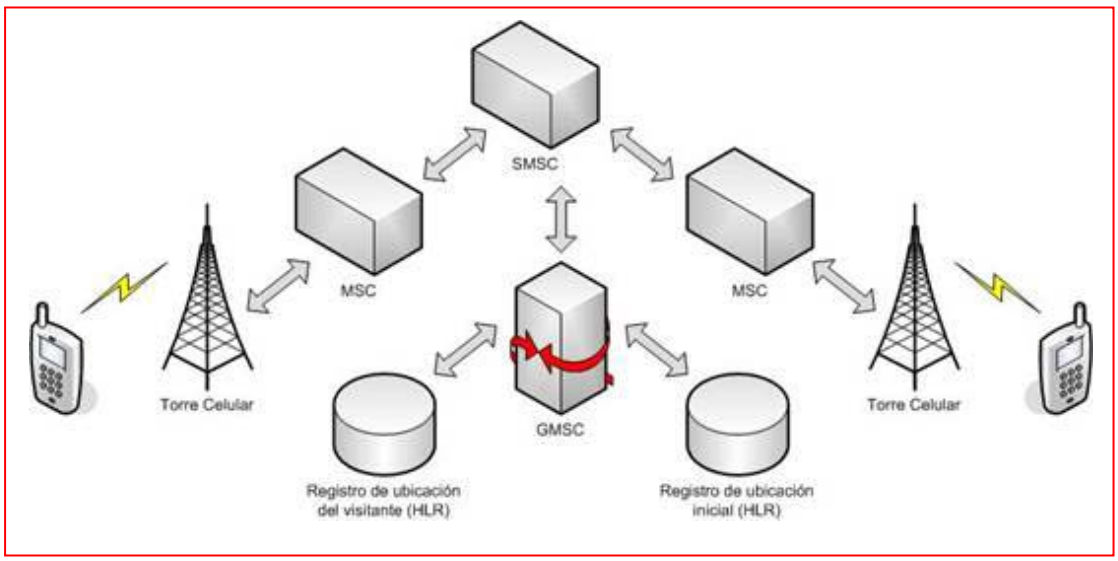


Ilustración 3-Arquitectura de red SMS. [8].

Como podemos observar, los mensajes pasan por uno o varios centros de distribución SMSC donde son almacenados durante un tiempo, no van directamente de un móvil a otro. Como curiosidad, cuando borramos un SMS almacenado en la tarjeta SIM, solo se modifica un byte por lo que es relativamente sencillo recuperar un mensaje borrado mediante programas de recuperación de datos. Y si no fuese posible,

hipotéticamente podríamos contactar con el operador de red para recuperarlo, aunque, obviamente por motivos de seguridad, no suele seguirse esta política en las operadoras. Aún así, el sistema de comunicaciones SMS no parece un sistema muy seguro por todo lo comentado anteriormente.

2.2.2. SERVICIO DE DIFUSIÓN CELULAR (CELL BROADCAST SERVICE)

CBS o SMS-CB [8] fue diseñado con el propósito de entregar mensajes a múltiples destinatarios en un área específica, es decir, los teléfonos móviles que se encuentran en la zona de influencia de una o más torres, a diferencia de un SMS o SMS Punto a Punto (SMS-PP), que se realizan de un usuario a otro, o de un usuario a unos pocos.

Las aplicaciones generales incluyen alertas de emergencia o de tráfico, servicios basados en la ubicación e información del operador, siendo su principal ventaja la velocidad de llegada a las personas, normalmente en unos pocos segundos.

La infraestructura, que podemos observar en la ilustración 4, cuenta con algunos elementos esenciales. El primero de ellos por supuesto es la *torre celular* o estación base, la cual se encuentra gobernada por el *controlador de estación base* (BSC). El nodo raíz de esta red es el *centro de difusión celular* (CBC), y actúa como servidor para todas las entidades externas de difusión.

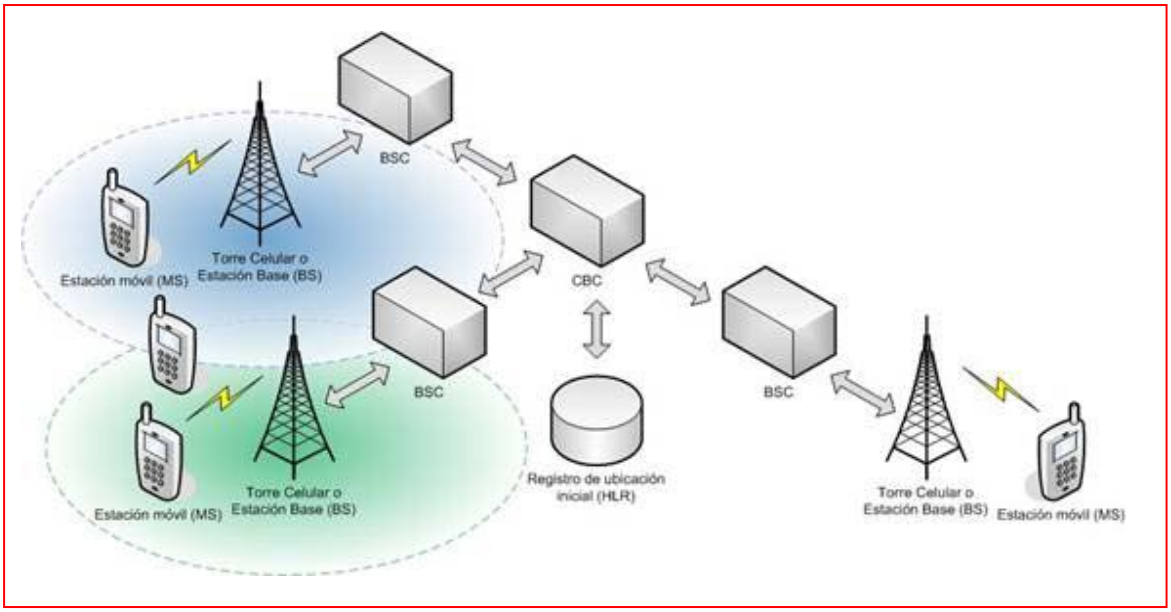


Ilustración 4- Arquitectura de red CBS. [8]

A diferencia de un SMS-PP, un mensaje de difusión es no confirmado, no existen reportes de entrega exitosa, sin embargo, poseen un canal dedicado para evitar, en la medida de lo posible, los atascos.

Un mensaje CBS contiene un total de 88 octetos, donde 82 de ellos corresponden al cuerpo del mensaje, equivalentes a 93 caracteres, pudiendo concatenar hasta 15 páginas. Su estructura es esquematizada en la siguiente figura.

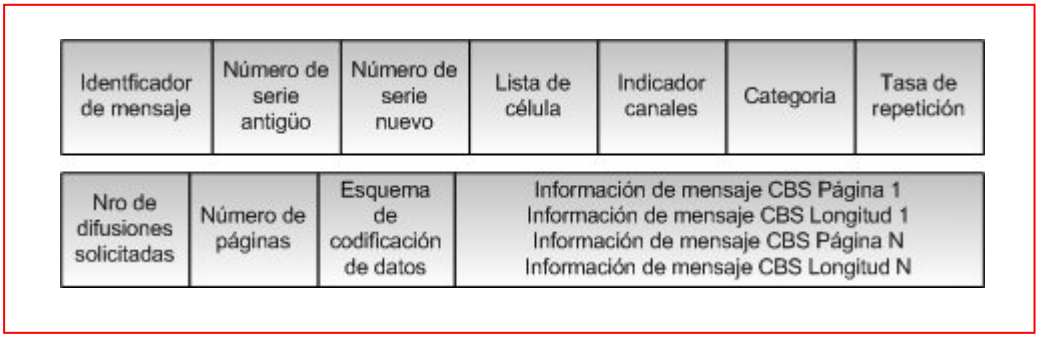


Ilustración 5 - Estructura de un mensaje CBS [8]

2.2.3. SERVICIO DE MENSAJE MULTIMEDIA (MULTIMEDIA MESSAGE SERVICE)

Debido a la creciente exigencia por parte de los usuarios, MMS [8] se ha constituido como una evolución más sofisticada del popular SMS, que además de texto plano debe ser capaz de transmitir distintos formatos de datos dentro de un mensaje multimedia (MM), a saber, audio, audio sintético, imágenes fijas, gráficos *bitmap*, gráficos vectoriales, vídeo, archivos con contenido dinámico, presentación y sincronización de medios y DRM (Direct Rendering Manager: es un componente de Direct Rendering Infrastructure (Infraestructura de Renderizado Directo o DRI), que provee a sistemas operativos tipo Unix, como Linux, FreeBSD, NetBSD, y OpenBSD de aceleración gráfica eficiente, especialmente para aplicaciones 3D). Cabe aclarar que dicha transmisión no es en tiempo real, sino del tipo “guardar y enviar”.

La variedad de datos a transferir hace que surjan problemas de distinta índole, por ejemplo, la diversidad de escenarios posibles, la incompatibilidad de formatos entre distintos dispositivos, el ancho de banda requerido, la complejidad en la configuración de los teléfonos, por nombrar sólo algunos. Todo esto impacta en la complejidad de las

especificaciones, llevadas a cabo por *Third-Generation Partnership Program* (3GPP [2]) en tres etapas:

1. Requerimientos (3GPP ***TS 22.140*** [3])
2. Funciones de sistema (3GPP ***TS 23.140*** [4])
3. Realización técnica

delegando la tercera al Open Mobile Alliance (OMA [5]), que a pesar de haberse comenzado en abril de 1999, todavía se siguen liberando versiones siendo la última de marzo de 2008 (3GPP ***TS 23.140*** [4] V6.15.0).

Algunas de las funcionalidades que debe presentar son:

- Administración de MM (creación, personalización, conversión, multiplicidad de formatos, priorización, hipervínculos, DRM, etc.)
- Envío y recepción de MM (envío desde el usuario, recepción automática y a pedido, *streaming*, concurrencia, etc.)
- Notificación y acuse de recibo (MM recibidos, envíos y entregas exitosas o fallidas, borrado, etc.)
- Direccionamiento (diferentes formatos de direcciones, múltiples destinatarios, esquemas MSISDN o correo electrónico, copia oculta, etc.)
- Administración y control de repositorio de red (persistencia de MM)
- Mensajes de error
- Perfiles de usuario (administración general, configurar ambiente multimedia, filtros, etc.)
- Seguridad
- Flexibilidad de facturación (a cargo del remitente, del destinatario, por longitud, tipo, número, entre otros, de MM)
- Interfaces externas (computadoras, PDA, etc.)

2 Web ***3GPP*** http://www.3gpp.org/

3 ***TS 22.140*** MMS; Stage 1: http://3gpp.org/ftp/Specs/html-info/22140.htm

4 ***TS 23.140*** MMS; Functional Description;Stage 2: http://www.3gpp.org/ftp//Specs/html-info/23140.htm

5 Web ***OMA*** http://www.openmobilealliance.org/

- Interconexión (con tecnologías de mensajes existentes, redes inteligentes, etc., dentro o fuera del ambiente móvil)
- *Roaming* (entre distintos operadores)
- Soporte para servicios específicos de un operador

La base de conectividad entre diferentes redes debe ser provista por el protocolo de Internet, este enfoque permite que los servicios de mensajería encontrados en ella sean compatibles con los de las redes 2G y 3G, la siguiente figura ilustra las relaciones existentes entre los diferentes elementos de la *arquitectura de la red MMS* (MMSNA: MMS Network Architecture).

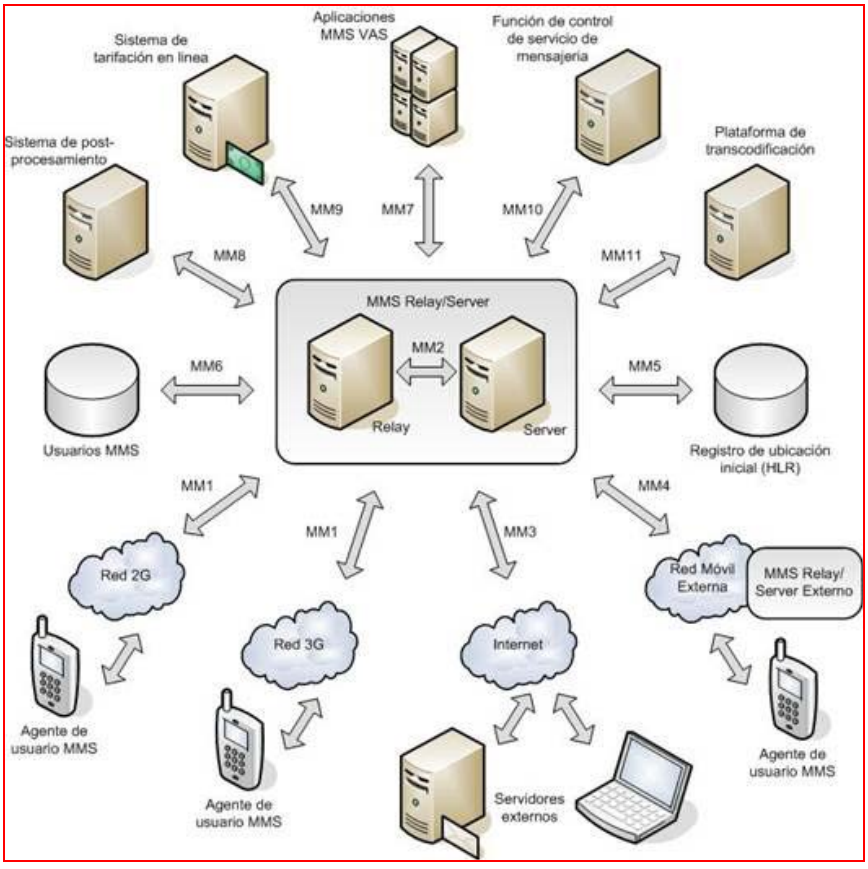


Ilustración 6 - Arquitectura de la red MMS [8].

El *Ambiente MMS* (MMSE) es la colección de elementos MMS bajo el dominio de un administrador, y suele existir más de una MMSE interconectada para el envío de MM.

En los extremos generalmente se encuentran los **agentes usuarios de MMS**, estos constituyen una capa de aplicación que brinda al usuario la posibilidad de manejar los MM, y normalmente se implementan para los teléfonos móviles.

El elemento aglutinante y central de la MMSE es la *MMS Relay/Server* responsable del almacenamiento, manejo y direccionamiento de los MM entrantes y salientes, junto con la transferencia entre distintos sistemas, encargada además de generar la tarificación tanto para la comunicación normal, como para las **aplicaciones de servicios de valor agregado** (VASA) correspondiente a los *proveedores VAS* (VASP).

Para que la interconexión sea exitosa se ha definido un conjunto de interfaces estándar para cada tipo de relación posible, denominadas en forma genérica con MMx existen un total de 11.

La estructura de un MM, ver la ilustración 7, sigue el mismo formato utilizado en el correo electrónico, consistente, fundamentalmente, en un encabezado y un cuerpo. El primero se rige por el ***RFC 822*** [6] y el segundo se corresponde con el formato MIME especificado en el ***RFC 2045-7*** [7], ya que suele contener varias partes y diferentes tipos de datos, asegurándose de esta manera compatibilidad, puesto que es un estándar ampliamente adoptado.

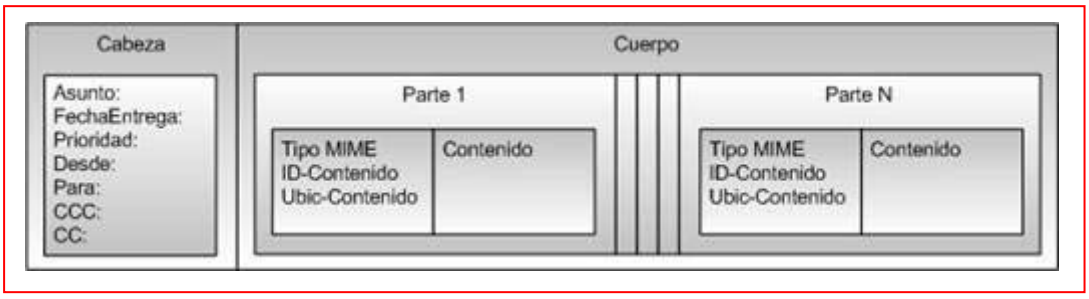


Ilustración 7 - Estructura de un mensaje MMS [8].

6 ***RFC 822*** STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES:disponible en: http://www.ietf.org/rfc/rfc0822.txt?number=822

7 ***RFC 2045*** Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies: disponible en http://www.ietf.org/rfc/rfc2045.txt

2.3. Nokia MMS Java Library

La librería Nokia MMS Java Library [25] facilita a los desarrolladores crear aplicaciones basadas en mensajes multimedia. La librería provee ejemplos de cómo crear, codificar, enviar y decodificar mensajes multimedia y las especificaciones de la interfaz de aplicaciones externas.

Los desarrolladores pueden aprender las funcionalidades básicas que son necesarias para desarrollar aplicaciones externas al MMSC. Las librerías también pueden usarse para propósitos comerciales. Los ejemplos que vienen con la librería explican las diferentes funcionalidades:

- Creación y codificación de mensajes MMS: el ejemplo muestra cómo crear y codificar un MMS con diferentes contenidos. La creación del mensaje se hace de acuerdo a la especificación ***WAP-209*** [8] MMS de encapsulación [32].
- Decodificación de mensajes: el ejemplo muestra cómo decodificar los MMS de acuerdo a la especificación WAP-209 MMS de encapsulación [32] y cómo extraer los contenidos multimedia del cuerpo del mensaje también.
- Enviar MMS: muestra cómo enviar mensajes MMS al MMSC. Se utiliza el protocolo HTTP 1.1 EIAF (implementación de Nokia de la interfaz MM7) cuando se envía un mensaje entre una aplicación y el centro de mensajería.

2.3.1. MM7

MM7 es una interfaz de servicio Web móvil estandarizada por el 3GPP. Los servicios Web han sido rápida y ampliamente aceptados en las tecnologías de integración de negocios, y en el entorno móvil actúa como puente entre servidores móviles y servidores de aplicaciones.

[8] ***WAP – 209*** **MMS Encapsulation Protocol:** Disponible en http://www.openmobilealliance.org/tech/affiliates/wap/wap-209-mmsencapsulation-20020105-a.pdf

Las interfaces de los servicios Web proveen a los operadores y desarrolladores la oportunidad de ofrecer nuevos componentes y rápido soporte con innovadores servicios móviles.

Las interfaces de servicios Web móviles están estandarizadas en muchos foros industriales tales como OMA (Open Mobile Alliance), 3GPP (Third Generation Partnership Project) y el ***W3C*** [9] (World Wide Web Consortium), y su implementación está basada en soluciones tecnológicas estandarizadas y ampliamente aceptadas: ***SOAP***[10] , ***XML*** [11], ***WSDL*** [12] (Web Services Description Languaje) y ***HTTP*** [13].

SOAP y XML han sido escogidos por el 3GPP como estándar para la arquitectura OSA (Open Services Architecture).

2.3.2. MMSC

El MMSC [21] o centro de mensajes multimedia (Multimedia Message Service Center) es similar en función a un centro de mensajes cortos SMSC, si bien presenta mayor capacidad y funciones más avanzadas.

El MMSC, en un sistema móvil GSM, debe comunicarse con la red núcleo de GPRS para el transporte de los mensajes, con el subsistema de red (NSS – VLR – HLR) para la tarificación y gestión de permisos del usuario, con un SMSC para remitir los avisos de mensajes entrantes y con Internet para poder enviar mensajes de correo electrónico. Además, se contempla la posibilidad de que el MMSC reconozca las características del terminal del usuario (tipo y tamaño de pantalla, capacidad de visualización de ciertos tipos de archivo y no de otros, etc.) y adapte automáticamente el contenido recibido al terminal para que se vea correctamente.

9 Web ***W3C*** disponible: http://www.w3.org/

10 Estándar ***SOAP***: http://www.w3.org/TR/soap/

11 Estándar ***XML***: http://www.w3.org/XML/

12 Estándar ***WSDL***: http://www.w3.org/TR/wsdl

En otro tipo de redes el método de transporte será distinto, pero el MMSC conservará la capacidad de adaptación del contenido, si está habilitada esta capacidad.

Este es un resumen de sus funciones:

- Recibir a través de la red núcleo GPRS los mensajes multimedia entrantes.
- Almacenar los mensajes entrantes hasta que puedan ser transportados hasta el usuario.
- Enviar al usuario los mensajes por la red GPRS cuando el usuario los descargue (normalmente, de forma automática a petición de su terminal).
- Recibir y redirigir hasta el MMSC destino los MMS salientes.
- Decidir si el tipo de contenido de los mensajes entrantes es válido para el terminal del usuario y, en su caso, procesarlo para que pueda mostrarse.
- Comunicarse con el HLR y VLR para verificar si el usuario tiene permiso para enviar mensajes y poder tarificarlos.

2.4. Plataformas de Desarrollo

2.4.1. DESARROLLO EN ESCRITORIO

Es la forma más común a la hora de programar un dispositivo de mano [6] y la que hemos utilizado a la hora de desarrollar la aplicación que se describe en este documento. Esta técnica consiste en realizar la aplicación a través de alguna herramienta ejecutada en un PC. Las herramientas pueden ser extensiones para los lenguajes y entornos habituales de los PC's o bien entornos de desarrollo específicamente creados para generar código ejecutable sobre los dispositivos de mano.

Dado que supondría un inconveniente tener que cargar en el dispositivo de cómputo móvil (DCM) el programa en desarrollo cada vez que se necesite ser probado, conviene contar con **emuladores**, como también veremos en el apartado de desarrollo.

13 Estándar ***HTTP***: http://www.w3.org/Protocols/

En ellos se puede crear la aplicación y depurarla sobre sus sistemas operativos. El desarrollo en este tipo de software es igual que el desarrollo en escritorio, solamente cambiando la compilación (cross compilation), que es la última parte del proceso.

Normalmente, el proceso de puesta a punto de las aplicaciones concluye con la generación del código de la aplicación que se instalará en dispositivo de mano. En este caso existen dos filosofías de trabajo: o bien el código generado es directamente ejecutable en la máquina, o bien se genera un código intermedio que al instalarse requerirá de un programa intérprete que lo traduzca y ejecute definitivamente.

Un aspecto fundamental a la hora de desarrollar una aplicación en un emulador, es el tener en cuenta que el emulador es una réplica del dispositivo. Debido a esto, hay veces que los programas realizados no funcionan como inicialmente esperamos una vez desarrollado y cargado en el dispositivo, como veremos en el apartado dedicado al desarrollo de la aplicación y la adaptación a la interfaz.

2.4.2. SUN: J2ME

Desarrollada por ***SUN*** [14] [6], ***J2ME*** [15] es el conjunto básico de herramientas que permite crear Java Applets y aplicaciones Java Standalone, para dispositivos con capacidades limitadas.

La configuración del J2ME consiste en un conjunto mínimo de APIs útiles de para desarrollar aplicaciones para un conjunto definido de dispositivos. Una interfaz de programación de aplicaciones o API (del inglés Application Programming Interface) no es más que el conjunto de funciones y procedimientos (o métodos, si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son muy importantes porque describen las funcionalidades más importantes requeridas para unos dispositivos determinados.

Existe una configuración estándar llamada Connected Limited Device Configuration o ***CLDC*** [16]. Ésta describe el conjunto de funcionalidades mínimas de los

[14] Sitio ***SUN***: http://www.sun.com/ y para desarrolladores java: http://java.sun.com/

[15] Sitio ***J2ME***: http://java.sun.com/javame/index.jsp

[16] Especificaciones ***CDLC***: http://java.sun.com/products/cldc/

dispositivos inalámbricos, acorde a su potencia y a sus características. Consiste en un resumen del conjunto de APIs básicas de construcción de aplicaciones para dispositivos móviles.

Las características de los API's básicos del CLDC son:

- Especifican aspectos de programación, como lenguaje utilizado y máquina virtual, y mínimos de hardware que requiere J2ME, como el procesador y el consumo.
- En lo que se refiere a limitaciones de Java, no permiten hacer ciertas operaciones matemáticas, eliminan métodos y limitan la capacidad de CLDC para manejar excepciones.
- En seguridad, se definen unas pautas del modelo sandbox , como son la carga de clases, la ejecución de APIs no autorizados, etc. De todas maneras, estas restricciones son de bajo nivel.

Existe la configuración ***MIDP*** [17] , que es un paso adelante en la configuración CLDC.

Los puntos destacables de MIDP [8] son los siguientes:

- **Riqueza en la interfaz de usuario**: altamente gráfica, intuitiva, optimizada para tamaño pequeño, distintos métodos de entrada (teclados de teléfonos, botones extra de navegación, pantallas sensibles al tacto, mini-teclados QWERTY).
- **Instalación local y ejecución local de aplicaciones**: pueden operar en redes intermitentemente conectadas, tanto en modo seguro como no seguro. Pueden persistir localmente los datos de las aplicaciones.
- **Amplia conectividad**: Soporte de estándares de conectividad establecidos: HTTP, HTTPS, datagramas, zócalos (*sockets)*, zócalos de servidor (*server sockets*), puerto serie. Servicios estándares como SMS Punto a Punto, donde el mensaje es generalmente entre dos móviles o entre un móvil y un

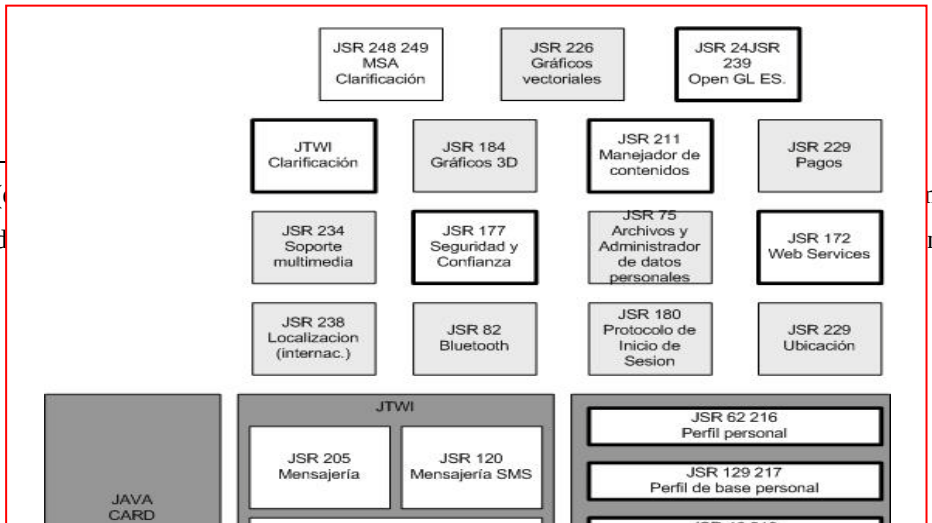
[17] Especificación ***MIDP***: http://java.sun.com/products/midp/

conjunto reducido de móviles, y SMS - Servicio de Difusión de Celdas (SMS-Cell Broadcast Service), donde un usuario puede enviar a una cantidad considerable de recipientes, todo esto a través de redes GMS-CDMA utilizando el paquete opcional WMA (Wireless Messaging API).

- **Funcionalidad multimedios y de juegos**: MIDP es una plataforma adecuada para aplicaciones de juegos y multimedios. La interfaz API de alto nivel se complementa con la adición de una de bajo nivel, que agrega funcionalidad específica para juegos, tales como *sprites*[18] y *capas superpuestas*, tomando las capacidades gráficas intrínsecas del dispositivo. También se da soporte a audio, secuencias de tono y archivos de tipo WAV. Utilizando el paquete opcional de API de medios móviles (Mobile Media API – MMAPI) se puede agregar a las aplicaciones MIDP el manejo de contenido multimedia avanzado.

- **Provisión a-través-del-aire**: Permite desplegar y actualizar las aplicaciones de forma dinámica y segura, utilizando una conexión inalámbrica.

- **Seguridad de red:** MIDP incorpora un conjunto de mecanismos de seguridad robusto que cumple con estándares abiertos y protege la red, las aplicaciones y los dispositivos móviles. HTTPS da soporte a SSL y a WTLS (capa segura de transmisión inalámbrica o *wireless transmission layer security*)



[18] ***sprites*** (
mapa de
bits dibujad
nales de
la CPU.

Ilustración 8- Diagrama de bloques constituyentes de J2ME [8]

2.4.3. MICROSOFT .NET

Microsoft.NET [6], es el conjunto de nuevas tecnologías en las que ***Microsoft*** [19] ha estado trabajando durante los últimos años para competir con la plataforma Java.

Esta plataforma se ha desarrollado con los siguientes objetivos:

- Mejorar sus sistemas operativos
- Mejorar su modelo de componentes COM+
- Obtener un entorno específicamente diseñado para el desarrollo y ejecución del software en forma de servicios. Estos pueden ser publicados por escrito o accesibles a través de Internet de forma independiente del lenguaje de programación, del modelo de objetos, del sistema operativo o del hardware utilizados. Este entorno se denomina Plataforma.NET, y a los servicios antes mencionados se les conoce por servicios web.

Para el desarrollo y ejecución de aplicaciones en este nuevo entorno, Microsoft proporciona un conjunto de herramientas conocidas como ***.NET Framework SDK*** [20],

[19] Web ***Microsoft***: http://www.microsoft.com/en/us/default.aspx

20 Framework de desarrollo disponible en:

que incluye compiladores de lenguajes como C#, Visual Basic.NET, Managed C++ y JScript.NET específicamente diseñados para crear aplicaciones para él.

2.4.4. OPENMOKO

OpenMoko es [26] un proyecto para crear una plataforma para smartphones usando software libre. Usa el núcleo Linux, junto con un entorno gráfico de usuario construido con el servidor X.Org, el toolkit GTK+ y el gestor de ventanas Matchbox. Está basado en el framework de OpenEmbedded y el sistema de paquetes ipkg.

OpenMoko se anunció en 2006 por sus fundadores: First International Computer (***FIC*** [21]). Los distintos modelos del teléfono son nombrados con las siglas GTA, que significa GNU Telephony Appliance. El 2 de abril del 2009, Openmoko canceló los teléfonos planeados y se concentra ahora en el actual FreeRunner y otros dispositivos.

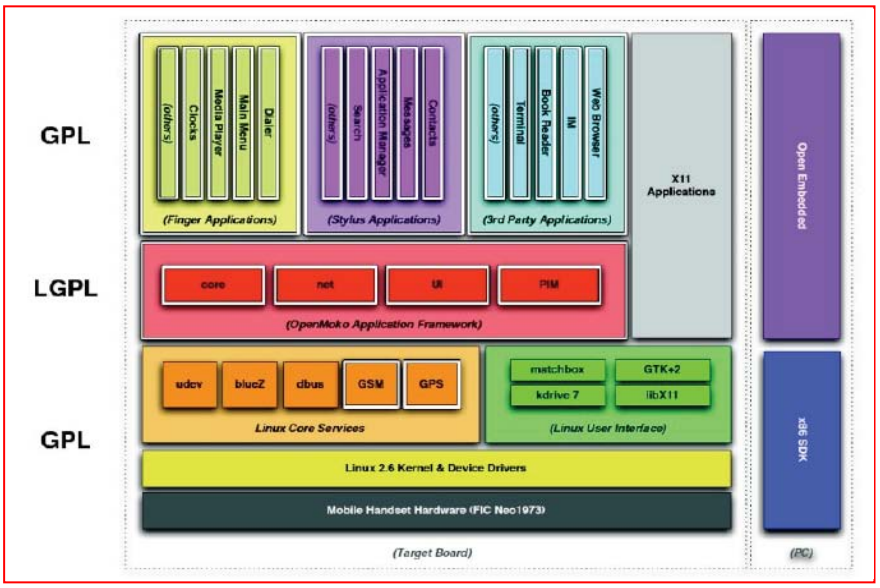


Ilustración 9 -Arquitectura de la plataforma de OpenMoko [26]

2.4.5. ANDROID

http://www.microsoft.com/downloads/details.aspx?FamilyID=fe6f2099-b7b4-4f47-a244-c96d69c35dec&displaylang=en

21 Web ***FIC*** disponible en: http://www.fic.com.tw/

Android [1] es un sistema operativo para dispositivos móviles y computadoras basado en el núcleo Linux. Inicialmente fue desarrollado por Google y luego por la ***Open Handset Alliance*** [22] (liderada por la propia ***Google*** [23]).

La presentación de la plataforma Android se realizó el 5 de noviembre de 2007 junto con la fundación Open Handset Alliance, un consorcio de 48 compañías de hardware, software y telecomunicaciones comprometidas con la promoción de estándares abiertos para dispositivos móviles.

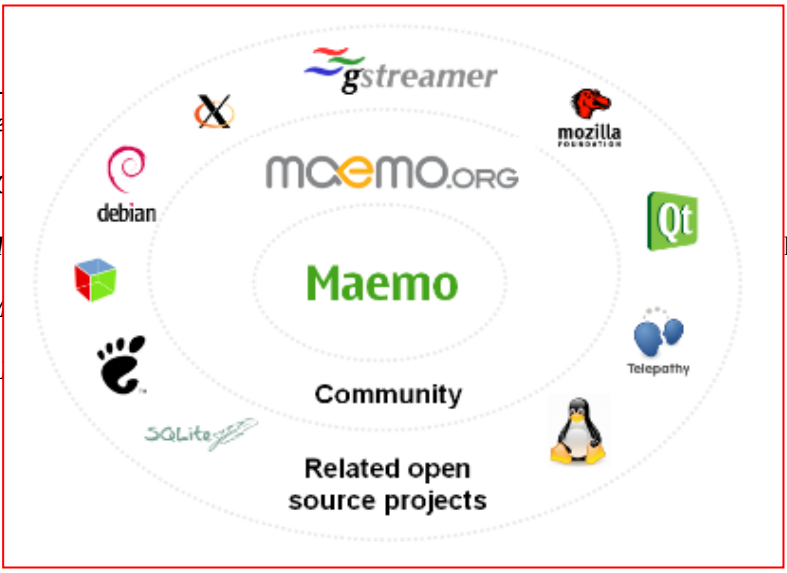
Esta plataforma permite el desarrollo de aplicaciones por terceros (personas ajenas a Google). Los desarrolladores deben escribir código gestionado en el lenguaje de programación Java a través de la SDK que proporciona Google. Una alternativa es el uso de la ***NDK*** [24] (Native Development Kit) de Google para hacer el desarrollo de las partes del código críticas en rendimiento en código nativo.

La mayoría del código fuente de Android ha sido publicado bajo la licencia de software Apache, una licencia de software libre y código fuente abierto. Google lanzó la última versión de Android, la Cupcake 1.5, en abril de 2009.

2.4.6. MAEMO

Maemo [25] es una plataforma software [14] desarrollada por ***Nokia*** [26] para *smartphones* e *Internet Tablets.* Está basado en el sistema operativo Linux. La plataforma comprende el sistema operativo Maemo y la SDK Maemo.

Maemo está basado casi en su totalidad en código libre y ha sido desarrollado en colaboración con otros proyectos de código libre como Linux kernel, Debian, y GNOME. Maemo está basada en Debian Linux y utiliza muchos de los componentes gráficos, frameworks y librerías del proyecto GNOME. Usa el gestor de ventanas y el Hildon (basado en GTK) como su GUI y framework de aplicaciones.



[22] Sitio web ***Ope***

[23] Sitio web de ***(***

[24] Software ***And*** html

[25] Sitio oficial ***M***

[26] Sitio Web de ***N***

Ilustración 10 - Componentes Maemo, imagen obtenida de la página oficial de Maemo

El intefaz de usuario en Maemo es similar a muchas interfaces de dispositivos de mano y está provista de una pantalla de inicio o "Home" que actúa como punto central desde el cuál se puede acceder a todas las aplicaciones y configuraciones.

La pantalla de Inicio se divide en áreas para lanzamiento de aplicaciones, una barra de menús y un área configurable por el usuario donde puede poner diversas aplicaciones integradas o externas: barra de búsqueda de Google, RSS reader…

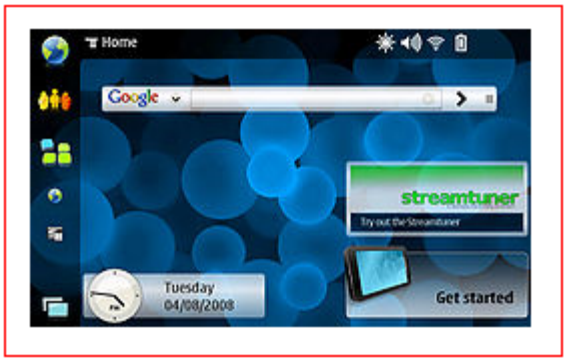


Ilustración 11 - Pantalla Home de Maemo

La intefaz usa tanto la pantalla táctil como el cursor de dirección y el botón select, con botones separados de atrás, menú y home. Está dotada con dos métodos de entrada de texto: reconocimiento de texto escrito a mano y teclado táctil en pantalla (o teclado deslizante en los modelos más recientes).

2.4.6.1. Algo de historia

El primer dispositivo de la serie de Nokia Tablet PC's [14] fue la Nokia 770 Internet Tablet que fue lanzada en Noviembre del 2005. La Nokia 770 Internet Tablet permitía el acceso a internet mediante una conexión WLAN.

El siguiente dispositivo fue la Nokia N800 Internet Tablet que llevaba la release 3 de Maemo. Se añadía a la funcionalidad de llamadas Skype/VoIP mediante WLAN y tenía una cámara integrada.

El siguiente paso fue la Nokia N810 que usaba Maemo 4. También tenía un teclado completo QWERTY que se extraía desde detrás de la pantalla táctil, un navegador basado en Mozilla, soporte GoogleTalk y GPS integrado.

Maemo 5 fue el mayor paso en la evolución de Maemo. Maemo 5 introdujo una interfaz finger-touch de usuario completamente rediseñado, teléfono móvil integrado y multitarea en la interfaz gráfica. Maemo 5 es la plataforma con la que se distribuye la Nokia N900.

2.4.6.2. Entorno de desarrollo

El desarrollo para Maemo se hace con Scratchbox Crosscompilation toolkit (licencia bajo GNU General Public License (GPL)), que es capaz de emular las arquitecturas INTEL y ARMEL, permitiendo compilar, probar y empaquetar el software en ambas, para posteriormente instalarlo en los dispositivos o distribuirlo en la comunidad.

Para desarrollar se utiliza el Maemo SDK. Existen diferentes releases del Maemo SDK. Estas releases incluyen las *rootstraps* normales (arm/armel e i386).

	Version	Codename	Build identifier	Release date	First device shipped with	Notes
OS2005	1.1	-	2.2005.45-1	November 2005	770	
			3.2005.51-13	December 2005		
			5.2006.13-7	April 2006		
OS2006	2.0	Mistral	0.2006.22-21	May 2006		Beta release
			1.2006.26-8	May 2006		
	2.1	Scirocco	2.2006.39-14	November 2006		
	2.2	Gregale	3.2006.49-2	January 2007	770	
OS2007	3.0	Bora	2.2006.51-6	January 2007	N800	

	3.1		3.2007.10-7	March 2007		
	3.2		4.2007.26-8	July 2007		
			4.2007.38-2	October 2007		SDHC corruption fix
OS2008	4.0	Chinook	1.2007.42-18	November 2007	N810	(N810 only)
			1.2007.42-19	November 2007		Kernel upgrade only (N810 only)
			1.2007.44-4	November 2007		Beta release (N800 only)
			2.2007.50-2	November 2007		
			2.2007.51-3	January 2008		NOLO upgrade only
	4.1	Diablo	4.2008.23-14	June 2008		Adds SSU support
			4.2008.30-2	August 2008		First SSU update
			4.2008.36-5	September 2008		
			5.2008.43-7	December 2008		
Maemo 5	5.0	Fremantle	-	October 2009	N900	Bundled community-supported Qt libraries
Maemo 6	6.0	Harmattan	-	- 2011		Bundled officially supported Qt libraries

Tabla 1 - Releases Maemo, tabla obtenida de la página oficial de Maemo

En la tabla 1 podemos observar las releases disponibles en la fecha de consulta de la web oficial de maemo (septiembre 2009).

Para el desarrollo de este proyecto se optó por la release Chinook, que era la versión más estable en el momento de la realización del proyecto (finales del 2007).

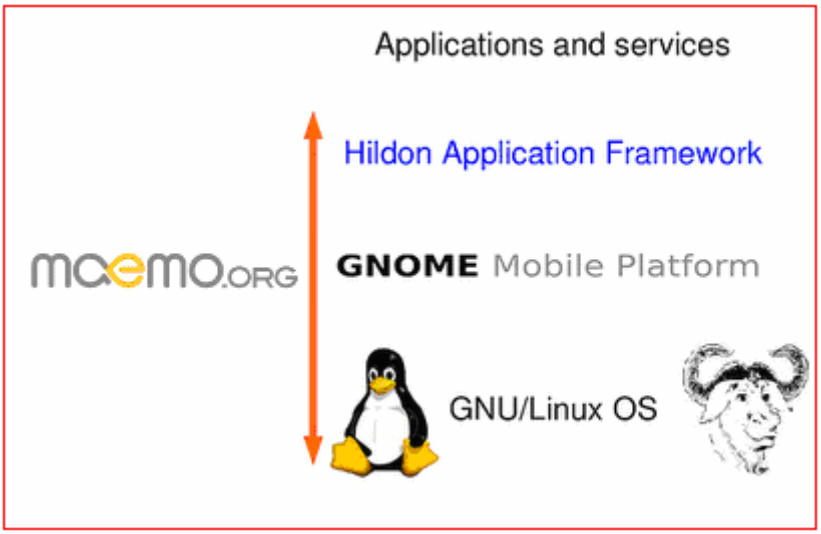


Ilustración 12 - Arquitectura release Chinook

2.4.7. MOBLIN

Moblin [27] [20] es un proyecto de código abierto enfocado al desarrollo de software orientado a dispositivos móviles para conectividad a Internet (MIDs) y nuevas clases de dispositivos como netbooks y nettops.

Amparado por la Fundación Linux, Moblin es un proyecto de sistema operativo de código abierto optimizado para ofrecer experiencias de contenidos ricos en Internet sobre dispositivos basados en el procesador Intel® Atom™ como los MIDs, netbooks/nettops, entretenimiento en coches (in-vehicle infotainment IVI), y sistemas embebidos.

Intel puso en marcha la web de Moblin.org en julio de 2007 y actualizó de manera significativa el sitio en abril de 2008 con el lanzamiento de la familia de procesadores Intel Atom en el Intel Developer Forum celebrado en Shanghai. En la web se puede encontrar también un SDK (kit de desarrollo). El sistema operativo Moblin 2 está especialmente diseñado para funcionar en procesadores Intel Atom, como los que podemos encontrar en un netbook.

En abril del 2009 Intel traspasó Moblin a la Linux Foundation.

Componentes principales

- **Moblin Image Creator (MIC):** permite a los desarrolladores el crear un sistema Linux adaptado para un dispositivo. Utilizando MIC, un desarrollador de una plataforma puede elegir qué componentes de Moblin quiere integrar en su dispositivo, construir el sistema, copiar todos los archivos necesarios a un pen drive USB y cargarlo en el dispositivo destino.
- **Kernel:** Adaptaciones del kernel de Linux específicas para la plataforma y muchos otros drivers para dispositivos.
- **UI Framework:** interfaz gráfica y el framework subyacente basado en GTK, que utiliza el framework de aplicaciones Hildon.
- **Política de gestión de energía:** extiende y mejora las capacidades de gestión de energía existentes en Linux.

[27] Siti web oficial ***Moblin*** y descarga de software: http://moblin.org/

- **Browser:** el navegador Moblin es un navegador web basado en tecnologías Mozilla con un interfaz manejado con el dedo y con integración con el UI del MID. El navegador Moblin soporta plugins como el Adobe Flash.
- **Multimedia:** sonido, vídeo y visualización de imágenes incluyendo los frameworks multimedia ***Helix*** [28] o ***GStreamer*** [29].
- **Linux Connection Manager:** Conexiones a Internet que pueden ser ampliadas a través de plugins que soporten múltiples tecnologías inalámbricas o de cable.

2.4.8. MOBLIN Y MAEMO

Intel Corporation y Nokia [31] anunciaron en julio del 2009 una relación a largo plazo para desarrollar una nueva clase de dispositivo de computación móviles basados en arquitectura Intel® y arquitecturas de chipsets que combinarán el rendimiento de ordenadores potentes con comunicaciones de banda ancha móvil de gran capacidad y conectividad ubicua a Internet.

Este esfuerzo también incluye desarrollo de tecnología y cooperación en varias iniciativas de software de código abierto para desarrollar tecnologías comunes. Dichas tecnologías podrán ser usadas en proyectos de la plataforma Moblin y Maemo, que ofrecerán sistemas operativos basados en Linux para esos futuros dispositivos de computación móvil.

Fomentar tecnologías comunes en los entornos de software. Moblin y Maemo ayudará a promover el desarrollo de aplicaciones compatibles para esos dispositivos, complementando el gran número de aplicaciones compatibles PC disponibles . Los proyectos de código abierto se dirigirán usando las mejores prácticas del modelo de desarrollo de código abierto.

2.5. Desarrollo de software

[28] Sitio web oficial ***Helix***: https://helixcommunity.org/

[29] Sitio web oficial ***GStreamer***: http://www.gstreamer.net/

En esta sección se mostrarán las actividades del proceso de desarrollo software y su organización en modelos de ciclo de vida, modelos que posteriormente aplicaremos al desarrollo de nuestra aplicación.

2.5.1. PROCESO DE DESARROLLO SOFTWARE

La idea de aplicar el concepto de proceso al desarrollo software proviene del campo de la fabricación, donde los procesos (pasos en la fabricación) están definidos y se controlan de manera continua. Este enfoque se puede aplicar al mundo del software para la gestión del proceso a nivel de proyecto y para mejorar las capacidades de los grupos de desarrollo. Según Cuevas Agustín [5], el proceso software establece el marco de trabajo tanto técnico como de gestión para poder aplicar los métodos, herramientas y personas a la tarea de desarrollo de software. La definición del proceso identifica los roles y las tareas específicas y establece medidas para el control de ejecución de cada paso.

La definición adecuada del proceso permitirá a una organización dedicada al desarrollo de software asegurar que cada elemento de trabajo se asigna apropiadamente y que se conoce su estado en cada momento. A su vez, esta definición oficial del proceso en la organización permitirá incorporar nuevos métodos que lo mejoren.

Un proceso definido permite que cada nuevo proyecto sea construido en base a la propia experiencia y la de sus predecesores. Los problemas recogidos a medida que se utiliza el proceso permiten identificar sus causas y corregirlos. De este modo, tanto el proceso como su definición y la infraestructura de soporte evolucionarán con la experiencia.

2.5.2. ACTIVIDADES DE GESTIÓN DEL PROCESO SOFTWARE

Las actividades de gestión están orientadas a controlar el desarrollo del proceso software y a corregir las desviaciones con respecto a los parámetros de calidad establecidos. Así, son tareas que se desarrollan a lo largo del ciclo de vida del software. Una vez obtenida la especificación de requisitos, su gestión se ocupa de fijar el marco de referencia, de carácter contractual, que determinará el desarrollo. Incluirá todas las actividades de control que mantengan la integridad y exactitud del acuerdo sobre los requisitos a medida que progrese el proyecto. Para ello, llevará a cabo el tratamiento y

control de las actualizaciones y cambios a los mismos garantizando su trazabilidad en el ciclo de vida.

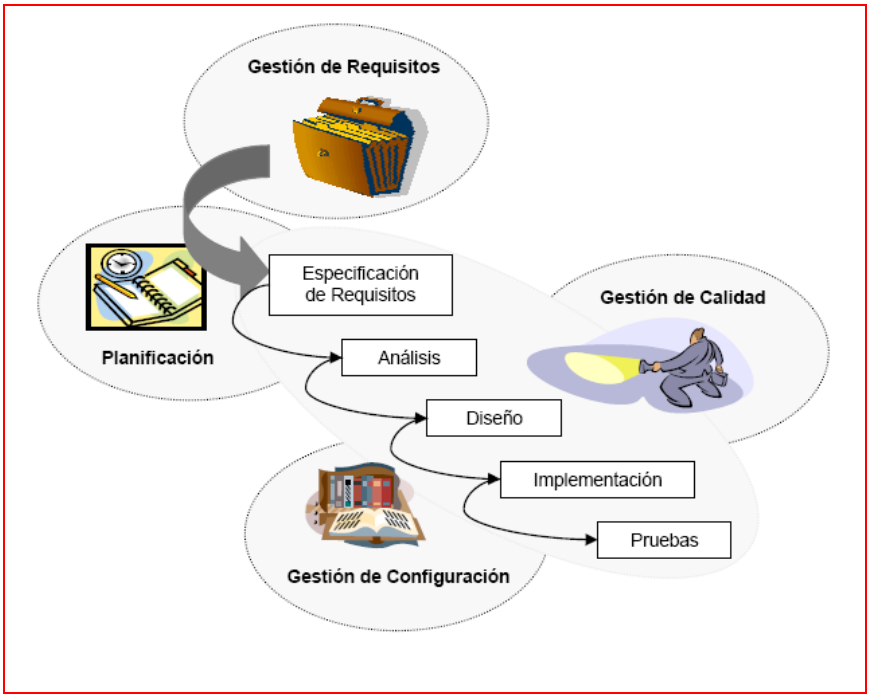


Ilustración 13 - Gestión del proceso software

La gestión de configuración del software es el medio que permite conocer, en todo momento, qué componentes y versiones, tanto de un producto como de sus elementos, son las correctas. De forma más concreta, la gestión de configuración es el proceso de identificar y definir los elementos de la configuración para:

- Controlar la liberación de versiones y cambios durante todo el ciclo de vida.
- Registrar e informar de su estado y de las peticiones de cambios.
- Verificar la corrección y acabado de los elementos.

Una vez establecidos los objetivos de calidad de un proyecto software y especificados aquellos procesos operativos y recursos necesarios para satisfacer dichos objetivos, la gestión de la calidad será el conjunto de actividades diseñadas para evaluar el proceso por el que se desarrollan los productos. Deberá garantizar que el sistema, componente o proceso cumple los requisitos especificados y cubre las necesidades o expectativas del cliente o usuario.

Por último, la planificación del proyecto consistirá en establecer la estructura temporal de las fases, actividades y tareas del proyecto, en función de los recursos de

que se disponga. El elemento de configuración es un conjunto software, y opcionalmente hardware, tratado como una unidad por la gestión de configuración seguimiento y control de evolución determinarán la necesidad de nuevas planificaciones que puedan alterar la fecha de finalización de proyecto.

Se recomienda al lector la referencia [5] para profundizar en las tareas de gestión del proceso software.

2.5.3. CICLO DE VIDA DEL SOFTWARE

El ciclo de vida software es el periodo que comienza cuando un producto software es concebido y termina cuando deja de estar disponible. El ciclo de vida se divide normalmente en fases que estructuran y organizan las etapas de concepción, desarrollo y mantenimiento. Un modelo de ciclo de vida es la descripción de las distintas formas de desarrollo de un proyecto, es decir, la orientación que debe seguirse para obtener, a partir de los requerimientos del cliente, sistemas que puedan ser utilizados por dicho cliente. A continuación se describirán el modelo de vida en cascada por ser éste el empleado en el transcurso de la aplicación de la que es objeto este documento.

2.5.3.1. Ciclo de vida en cascada

El ciclo de vida en cascada es el ciclo más simple. Históricamente, fue el primer ciclo en aparecer. La vida de los sistemas pasa por una serie de fases consecutivas y separables. Las fases se desarrollan en secuencia y sólo una vez, aunque puede haber iteraciones dentro de cada fase. En la siguiente figura se esquematiza el ciclo de vida en cascada con cada una de sus fases.

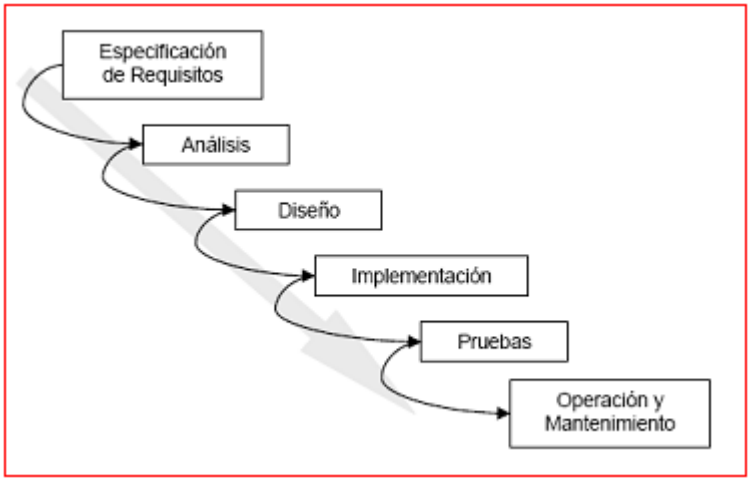


Ilustración 14 - Ciclo de vida en cascada

El modelo de ciclo de vida se escoge en la planificación inicial del proyecto. Cada proyecto se estructura en actividades, que serán previamente planificadas y agrupadas en fases.

Además, los productos deben ser terminados y entregados, de acuerdo al plan, y funcionar correctamente. Todo ciclo de vida debe incluir las fases incluidas en la siguiente tabla.

FASES CICLO DE VIDA		
FASE	**DEFINICIÓN**	**ENTREGABLE**
UR	Definición de Requerimientos de Usuario	URD
SR	Definición de Requerimientos del Software	SRD
AD	Diseño de la Arquitectura	ADD
DD	Diseño Detallado y codificación	DDD, SUM, CODE
TR	Transferencia del software	STD
OM	Operación y mantenimiento	PHD

Tabla 2 - Fases del ciclo de vida

Las primeras cuatro fases finalizan con una revisión. Estas fases han de aparecer siempre, independientemente del tamaño, tipo, software empleado, o equipo de desarrollo, ya sea propio o externo. El ciclo de vida del software empieza realmente con la entrega del Documento de Requisitos de Usuario (URD, User Requirements Document) a los desarrolladores. Por lo tanto, la revisión del URD es la primera

actividad del ciclo de vida. Junto con esta revisión, ha de producirse la entrega de un plan de gestión de proyecto, incluyendo una planificación y una estimación de costes para el proyecto. Los entregables de cada fase deben ser revisados y aprobados antes de proceder con la siguiente fase. Hay seis hitos que indican el progreso del ciclo de vida:

- Aprobación del URD.
- Aprobación del Documento de Requisitos de Software (Software Requirements Document: SRD).
- Aprobación del Documento de Diseño Arquitectónico (Architectural Design Document: ADD).
- Aprobación del Documento de Diseño Detallado (Detailed Design Document: DDD), Manual de Usuario del Software (Software User Manual: SUM) y Código, así como confirmación de disponibilidad para las pruebas de aceptación provisional.
- Entrega del Documento de Transferencia del Software (Software Transfer Document: STD) y confirmación de aceptación provisional.
- Entrega del Documento Histórico del Proyecto (Project History Document: PHD) y confirmación de aceptación final.

A continuación se describen las fases del ciclo de vida del software.

Definición de requisitos del usuario En esta fase se trata de plantear el problema que deberá ser solucionado mediante un sistema software. Para ello, se realizarán entrevistas con el cliente, de forma que se obtienen los requisitos que debe tener el sistema para resolver sus necesidades.

Definición de requisitos del software Esta es la fase de análisis del problema. A partir de las necesidades que han originado la puesta en marcha del proyecto, se trata de comprender qué es lo que tiene que realizar el sistema en su conjunto. Se analizará el entorno en el que se desenvolverá el sistema y la manera en que este interactuará con él.

Diseño de la arquitectura Esta fase corresponde al planteamiento de la solución. El sistema se descompone en un conjunto de subsistemas, de manera que se simplifica el sistema global. Se definen las relaciones que tiene cada uno de los subsistemas con los demás.

Diseño detallado y codificación En esta fase se refina el diseño realizado en la fase anterior. Además se realiza la implementación del sistema, se valida y se verifica, de manera que el sistema realizado cumpla con los requisitos definidos en las fases iniciales.

Transferencia del software En esta fase, el sistema es transferido a su entorno de explotación. Puede haber un periodo de enseñanza para que el usuario conozca la manera de utilizar el sistema.

Operación y mantenimiento Esta suele ser la fase más larga del ciclo de vida. Transcurre durante el uso del sistema en su entorno de explotación. Además tendrán lugar actuaciones de mantenimiento y mejora del sistema.

2.6. Usabilidad en aplicaciones

2.6.1. CONCEPTO

La usabilidad es la disciplina que estudia la forma de diseñar aplicaciones para que los usuarios puedan interactuar con ellos de la forma más fácil, cómoda e intuitiva posible. Dado que la aplicación que se está desarrollando está claramente orientada al usuario se seguirán, en el grado en que sea posible, los diez principios básicos de la usabilidad, enunciados por el gurú en esta materia, Jakob Nielsen. Estos principios son [24]:

Visibilidad del estado del sistema para los usuarios El sistema debe mantener a los usuarios informados en todo momento sobre lo que está sucediendo, utilizando para ello sistemas adecuados en tiempos razonables.

Adecuación entre el sistema y el mundo real El sistema debe hablar el lenguaje de sus usuarios con palabras y conceptos comprensibles por ellos. Se deben evitar, por tanto, términos orientados al sistema. Son necesarias las convenciones del mundo real y sobre todo, hacer que la información aparezca en un orden lógico.

Libertad y control por parte del usuario Es necesario implementar un mecanismo de hacer / deshacer, de esta manera el usuario no se verá obligado a buscar puertas de emergencia cuando piense que ha cometido un error.

Consistencia y estándares Los usuarios no tienen que preguntarse si diferentes palabras, situaciones o acciones significan lo mismo. Es necesario, por tanto, seguir los estándares de plataforma.

Prevención de errores En lugar de buenos y vistosos mensajes de error es mejor tener un buen diseño, que sea cuidadoso con los errores en el primer intento.

Reconocimiento antes que recuerdo Las instrucciones de uso del sistema deben encontrarse siempre visibles o fácilmente recuperables cuando sean necesarias.

Flexibilidad y eficiencia en el uso El sistema tiene que estar preparado para usuarios expertos e inexpertos de igual manera.

Diseño estético y minimalista Las alertas o mensajes que se muestren no deben contener información irrelevante o innecesaria. Cada unidad de información extra compite con la información realmente relevante, minimizando su visibilidad frente al resto.

Ayuda a los usuarios a reconocer, diagnosticar y recuperarse de los errores Los mensajes de error deben estar expresados en un lenguaje entendible para el usuario, describiendo de forma precisa el problema y proporcionando una solución clara.

Ayuda y documentación Es necesario dotar al sistema de un módulo de ayuda y documentación. Cualquier información debe ser fácil de buscar, estar centrada en la acción de usuario, proporcionar una lista de pasos a seguir y no ser demasiado extensa.

2.6.2. USABILIDAD EN APLICACIONES PARA DISPOSITIVOS MÓVILES

2.6.2.1. Entono dinámico

En un terminal móvil, el entorno en el que un usuario se desenvuelve es cambiante, dinámico. El usuario puede estar distraído o tener prisa, por lo que la estructura de la aplicación tiene que ser muy simple y hay que evitar los pasos innecesarios.

Por otro lado, la tarea que está realizando el usuario puede interrumpirse por pérdida de cobertura, por una llamada entrante o por una simple distracción. Por lo tanto, el diseño debería permitir recuperar el proceso en curso tras la interrupción.

2.6.2.2. Heterogeneidad de dispositivos

Los dispositivos en sí tienen unas características físicas que afectan negativamente a la usabilidad de una aplicación, como por ejemplo:

Tamaño de la pantalla

Tamaño de las teclas

Dificultad para escribir texto

2.6.2.3. Criterios fundamentales de diseño

2.6.2.3.1. Escribir es difícil

Aunque sea con escritura predictiva, esta tarea resulta difícil. Siempre es preferible la selección de opciones a la escritura, aunque lleve más pasos. Si no hay más remedio, es importante tener controlado el formato de entrada.

Se recomienda informar por defecto campos siempre que sea posible predecir el valor más probable de entrada. En este sentido, es muy interesante la funcionalidad de guardar formularios o ventanas ya rellenadas.

2.6.2.3.2. Navegación

La navegación, debido al entorno dinámico comentado anteriormente, debe ser intuitiva, fácil de seguir. Asimismo, si el usuario interrumpe el proceso por la distracción que sea, el programa debe guardar su estado para permitirle continuar cuando lo desee.

Es importante evitar, en la medida de lo posible, los pasos hacia atrás, ofreciendo una escapatoria en la navegación cuando el usuario decida cancelar la acción que está llevando acabo.

2.6.2.3.3. ¿Versión genérica?

Las limitaciones de los terminales móviles implican que el modelo de diseñar una versión genérica que funcione bien para todos, para el caso del teléfono móvil, llevará a que todos tengan una experiencia de usuario inferior a lo aceptable. Es decir, que el diseño genérico es malo para todos. Por lo tanto, es muy recomendable personalizar las aplicaciones, adaptando la interfaz al dispositivo/navegador de cada usuario, para explotar al máximo sus características. Es por este motivo que la optimización de la aplicación N800 MMS se hará para un dispositivo en concreto, el N800, aunque pueda ejecutarse en otros dispositivos,

3. ANÁLISIS FUNCIONAL

3.1. OBJETIVO

El objetivo de este capítulo es la elaboración del análisis funcional de la primera versión de la aplicación **MMS N800 UC3M.**

El capítulo se estructura en los siguientes apartados:

- Requisitos de usuario
- Casos de uso
- Interfaz de usuario

3.2. REQUISITOS

El departamento de Ingeniería telemática de la Universidad Carlos III de Madrid, para su asignatura Laboratorio de Ingeniería de Servicios en sus créditos de prácticas precisa de una aplicación de mensajería multimedia (MMS) que pueda ejecutarse en el entorno de producción descrito en el apartado 3.3 para que los alumnos de dicha asignatura puedan poner en práctica los conocimientos teóricos obtenidos sobre mensajería multimedia.

Dada la naturaleza docente de la aplicación, el aplicativo debe ajustarse a los requerimientos específicos de conocimiento del perfil del alumno de la asignatura. Igualmente, la interfaz de usuario debe ser intuitiva y sencilla, no penalizando por ello la funcionalidad requerida.

Otro requisito imprescindible es que, dado que las prácticas de la asignatura LIS son en el segundo cuatrimestre, el proyecto debe concluir antes de dicha fecha para que la aplicación pueda servir de material docente.

3.2.1. ESPECIFICACIÓN DE REQUERIMIENTOS

A continuación vamos a pasar a describir con detalle los requisitos que la aplicación de mensajería multimedia debe cumplir.

RU-000001- Recepción y envío de mensajes multimedia: La aplicación debe ser capaz de generar y recibir mensajes que contengan texto, imagen, sonido y video como adjuntos al mensaje de texto.

RU-000001: Recepción y envío de mensajes			
Author:		Kind:	Requisito de usuario
Version:	1.0	State:	Proposed
Priority:	High	Complexity:	High
Stability:	Normal	Verifiable:	True
Goal:		Permitirá el envío y recepción de MMS	
Related Requirements:			
Description:		Permitirá el envío y recepción de mensajes multimedia que contengan texto, imagen, video y sonido.	

Tabla 3 - RU-000001: Recepción y envío de mensajes

RU-000002- Adecuación al temario: La aplicación debe ser compatible con Java MMS Library y el Nokia MMSC External Application Interface Emulator, herramientas de trabajo de los alumnos de prácticas de la asignatura.

RU-000002: Adecuación al temario			
Author:		Kind:	Requisito de usuario
Version:	1.0	State:	Proposed
Priority:	Medium	Complexity:	Medium
Stability:	Normal	Verifiable:	True
Goal:		La aplicación debe ser compatible con las herramientas que utilizan los alumnos en prácticas.	
Related Requirements:		RU-000003: Adaptación al terminal de prácticas	
Description:		La aplicación debe ser compatible con las librerías y la central Nokia MMSC.	

Tabla 4 - RU-000002: Adecuación al temario

RU-000003- Adaptación al terminal de prácticas: La aplicación debe ser compatible con el terminal de producción designado para las prácticas: Nokia N800 Tablet PC. La adecuación al terminal no implica una solución propietaria ni específica para el terminal, se recomienda aprovechar el carácter "open source" del mismo.

RU-000003: Adecuación al terminal de prácticas			
Author:		Kind:	Requisito de usuario
Version:	1.0	State:	Proposed
Priority:	Medium	Complexity:	Medium
Stability:	Normal	Verifiable:	True
Goal:		La aplicación debe ser compatible con el terminal de prácticas.	
Related Requirements:			
Description:		La aplicación debe ser compatible con el terminal de prácticas (N800) aunque es deseable una solución no específica.	

Tabla 5 - RU-000003: Adecuación al terminal de prácticas

RU-000004- Nivel de trazado de error visible para los desarrolladores: La aplicación debe proveer de un sistema de logs con interfaz de consulta en el dispositivo.

RU-000004: Sistema de logs			
Author:		Kind:	Requisito de usuario
Version:	1.0	State:	Proposed
Priority:	Medium	Complexity:	Medium
Stability:	Normal	Verifiable:	True
Goal:		La aplicación debe contar con un sistema visible de logs.	
Related Requirements:			
Description:		Los logs deberán ser visibles dentro de la propia aplicación así como en formato texto en un archivo almacenado en el dispositivo.	

Tabla 6 - RU-000004: Sistema de logs

RU-000005- Interfaz gráfica amigable: La aplicación debe tener un interfaz amigable e intuitiva con la que un usuario habituado a manejar herramientas de mensajería en terminales móviles se sienta familiarizado.

RU-000005: Interfaz simple intuitiva			
Author:		Kind:	Requisito de usuario
Version:	1.0	State:	Proposed
Priority:	High	Complexity:	Medium
Stability:	Normal	Verifiable:	True
Goal:		La aplicación debe contar con una interfaz simple e intuitiva.	
Related Requirements:			
Description:		Se buscará un diseño semejante al de las aplicaciones convencionales de mensajería con las que el usuario inexperto se encuentra familiarizado.	

Tabla 7 - RU-000005: Interfaz simple intuitiva

RU-000006- Sistema de gestión de mensajes: La aplicación debe ser capaz de almacenar los mensajes recibidos y permitir al usuario gestionar estos recursos. Guardar mensajes en elaboración, eliminar mensajes, etc.

RU-000006: Gestión de mensajes			
Author:		Kind:	Requisito de usuario
Version:	1.0	State:	Proposed
Priority:	High	Complexity:	Medium
Stability:	Normal	Verifiable:	True
Goal:		La aplicación debe contar con un sistema de gestión de mensajes.	
Related Requirements:			
Description:		Mediante este sistema el usuario podrá gestionar los mensajes: crear, guardar, eliminar, consultar, etc.	

Tabla 8 - RU-000006: Gestión de mensajes

RU-000007- Sistema de alertas: La aplicación debe notificar al usuario la recepción de nuevos mensajes deforma visual y auditiva.

RU-000007: Notificación de mensajes			
Author:		Kind:	Requisito de usuario
Version:	1.0	State:	Proposed
Priority:	High	Complexity:	Medium
Stability:	Normal	Verifiable:	True
Goal:		La aplicación debe contar con un sistema de notificación de mensajes.	
Related Requirements:			
Description:		Cuando el dispositivo reciba un nuevo mensaje deberá notificarlo al usuario mediante una alerta gráfica.	

Tabla 9 - RU-000007: Notificación de mensajes

3.2.2. REQUISITOS DESEABLES

Como requisitos deseables, tanto el API de programación como la configuración para el despliegue deben ser lo suficientemente flexibles como para permitir la modificación del entorno de producción, sujeto a las necesidades de formación y sensible a las diferentes configuraciones que el departamento determine. Es por tanto que sería deseable una solución multiplataforma.

3.2.3. ENTORNO DE PRODUCCIÓN

El entorno de producción propuesto para las prácticas de la asignatura consiste en una red de terminales Nokia N800 interconectados mediante una red wifi que interactúen entre ellos o con un centro de mensajería externo, como puede ser el MMSC de Nokia, desplegado en un soporte con interfaz wireless.

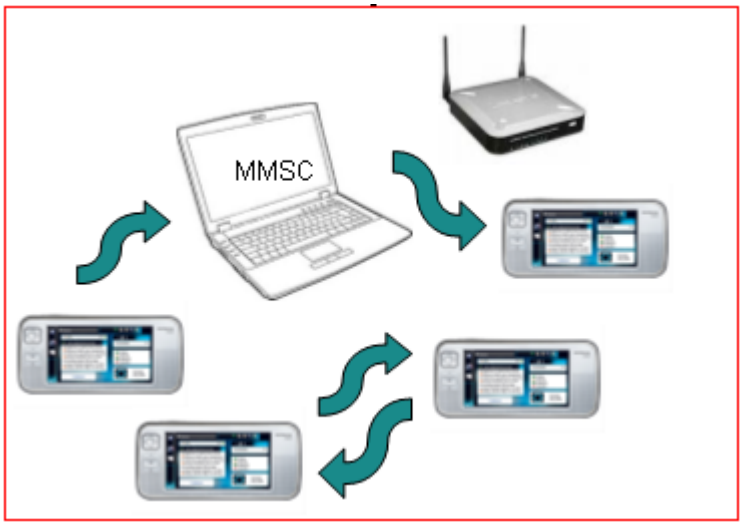


Ilustración 15 - Versión básica del entorno de producción

De forma alternativa al entorno inicial sería deseable un entorno multiplataforma en el que varios dispositivos con diferentes SSOO puedan interactuar.

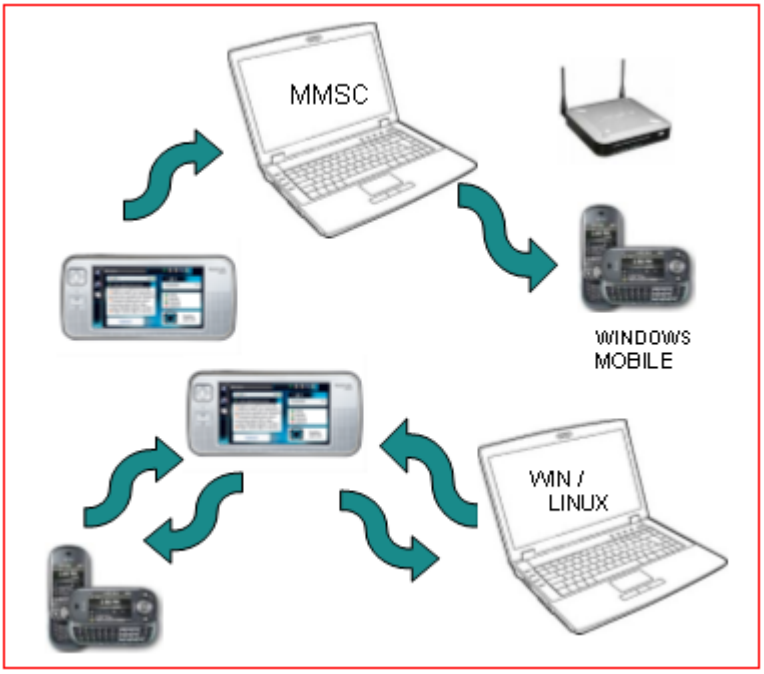


Ilustración 16 - Versión completa del entorno de producción

3.3. CASOS DE USO

Los casos de uso representados para el Sistema en la Fase de Análisis son:

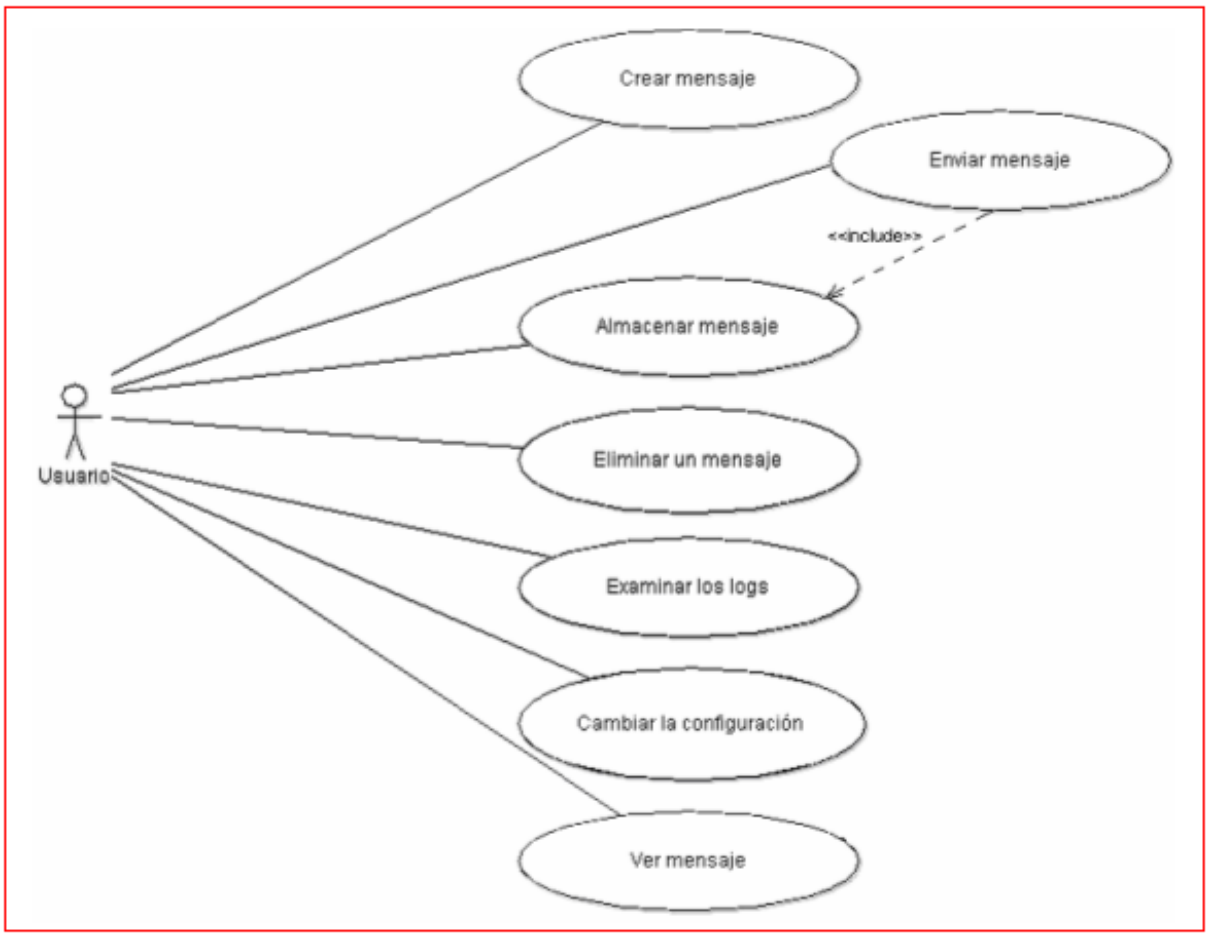


Ilustración 17 - Casos de uso en fase de análisis

Se ofrece a continuación una descripción más en detalle de los casos de uso representados en la figura anterior. Para cada caso de uso se ofrece una descripción del mismo, los actores que están involucrados en él, las precondiciones (estado del sistema que se tiene que cumplir para que el caso de uso se pueda instanciar), las post-condiciones (estado del sistema una vez instanciado el caso de uso) y el escenario básico (pasos principales del caso de uso ordenados).

3.3.1. CREACIÓN DE MMS

Nombre	Creación MMS.
Descripción	El sistema ha de permitir insertar los datos de un nuevo MMS: texto, video, sonido y/o imagen así como los campos de dirección de envío y asunto.
Actores	Usuario.
Precondiciones	Programa arrancado.
Postcondiciones	Sistema inalterado.
Escenario	Abrir pantalla de Nuevo mensaje. Introducir la información del mensaje en los contenedores habilitados a tal efecto en la pantalla correspondiente.

Tabla 10 - Caso de uso - Creación MMS

3.3.2. ENVÍO DE MMS

Nombre	Envío de MMS.
Descripción	El sistema ha de permitir el envío de un MMS en creación o almacenado en el dispositivo. Tras el envío el mensaje debe ser almacenado como elemento enviado. Si el mensaje ya existía deberá ser trasladado de contenedor.
Actores	Usuario.
Precondiciones	Existencia de conexión (inalámbrica) habilitada en el dispositivo y mensaje en creación o almacenado existente.
Postcondiciones	Mensaje enviado y almacenado.
Escenario	Enviar el mensaje a través de la conexión de red existente. Almacenar el mensaje enviado en la carpeta correspondiente. Modificación del fichero de índices de MMS del sistema de gestión de ficheros.

Tabla 11 - Caso de uso - Envío de MMS

3.3.3. ALMACENAR MMS

Nombre	Almacenamiento de MMS.
Descripción	El sistema ha de permitir almacenar un nuevo MMS.
Actores	Usuario.
Precondiciones	Existencia de memoria libre en el dispositivo para el almacenamiento y existencia de mensaje en creación.
Postcondiciones	Mensaje almacenado.
Escenario	Almacenar el mensaje enviado en la carpeta correspondiente Modificación del fichero de índices de MMS del sistema de gestión de ficheros.

Tabla 12 - Caso de uso - Almacenamiento de MMS

3.3.4. VER UN MMS

Nombre	Ver un MMS.
Descripción	El sistema ha de permitir visualizar un MMS existente en cualquiera de las bandejas o contenedores. La visualización debe incluir previsualización de imágenes, de video, de los campos textuales y escuchar los archivos de sonido.
Actores	Usuario.
Precondiciones	Existencia de reproductor de vídeo integrado en el dispositivo y reproductor de sonido.
Postcondiciones	El sistema permanece inalterado.
Escenario	Abrir pantalla de mensaje existente Reproducción de video. Reproducción de sonido.

Tabla 13 - Caso de uso - Ver MMS

3.3.5. ELIMINAR UN MMS

Nombre	Eliminar MMS.
Descripción	El sistema ha de permitir eliminar un MMS existente.
Actores	Usuario.
Precondiciones	Existencia de un mensaje.
Postcondiciones	Mensaje eliminado.
Escenario	Eliminar el mensaje de la carpeta correspondiente. Modificación del fichero de índices de MMS del sistema de gestión de ficheros.

Tabla 14 - Caso de uso - Eliminar MMS

3.3.6. VISUALIZACIÓN DE LOGS

Nombre	Visualización de logs.
Descripción	El sistema ha de permitir visualizar los logs de la aplicación que almacenan y registran toda la actividad de la aplicación.
Actores	Usuario.
Precondiciones	Existencia de espacio en el dispositivo para almacenamiento de los logs.
Postcondiciones	El sistema permanece inalterado.
Escenario	Abrir pantalla de visualización de logs.

Tabla 15 - Caso de uso - Visualización de logs

3.3.7. CONFIGURACIÓN DEL DISPOSITIVO

Nombre	Configuración de dispositivo.
Descripción	El sistema ha de permitir configurar el puerto de escucha del hilo de recepción y la dirección de envío de los mensajes.
Actores	Usuario.
Precondiciones	Existencia de conexión (inalámbrica) habilitada en el dispositivo.
Postcondiciones	El hilo de escucha se reinicia sin el reinicio de la aplicación.
Escenario	Abrir pantalla de configuración. Reinicio del hilo de escucha con la nueva configuración.

Tabla 16 - Caso de uso - Configuación de dispositivo

3.4. INTERFAZ DE USUARIO. DISEÑO DE PANTALLAS.

En el diseño de las pantallas del programa debe primar la simplicidad. Puesto que las aplicaciones SMS y MMS son interfaces de usuario ampliamente conocidas hoy en día, adoptaremos las convenciones de los aplicaciones existentes en el mercado.

A continuación podemos ver un diseño aproximado de las pantallas de la aplicación así como una breve descripción de sus funcionalidades.

3.4.1. DISEÑO BÁSICO

El diseño de las pantallas se ha formado en base a cuatro conceptos clave:

1.- Software libre: El *alma mater* del proyecto.

2.- Universidad Carlos III: la idea de la que surge el proyecto.

3.- Mensajería: el objetivo final del proyecto.

4.- La comunicación y el humo: El diseño de fondo de las pantallas surge de la idea de que un mensaje escrito es, con respecto a la comunicación verbal, como una fotografía de humo.

Ilustración 18 - Imagen de fondo de las pantallas

3.4.2. INTERFAZ DE ENTRADA

La interfaz de entrada estará provista de los botones de acceso a las diferentes bandejas estándar: bandeja de entrada, donde se almacenan los MMS recibidos, bandeja de salida, donde se almacenan los mensajes enviados, nuevo MMS, para escribir un nuevo mensaje y Borradores, donde se almacenan los mensajes creados, pero no enviados.

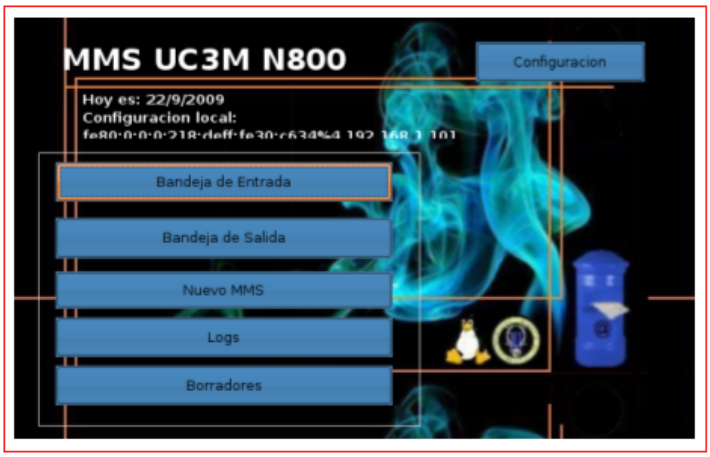


Ilustración 19 - Pantalla Principal

En la interfaz de entrada también aparecerán los botones de acceso a la Configuración del servicio y el botón de acceso a los Logs de la aplicación.

Asimismo, junto al botón de configuración se ubica el área de notificación de recepción de mensajes, en dicho área se mostrará un sobre amarillo a modo de alerta de mensaje recibido.

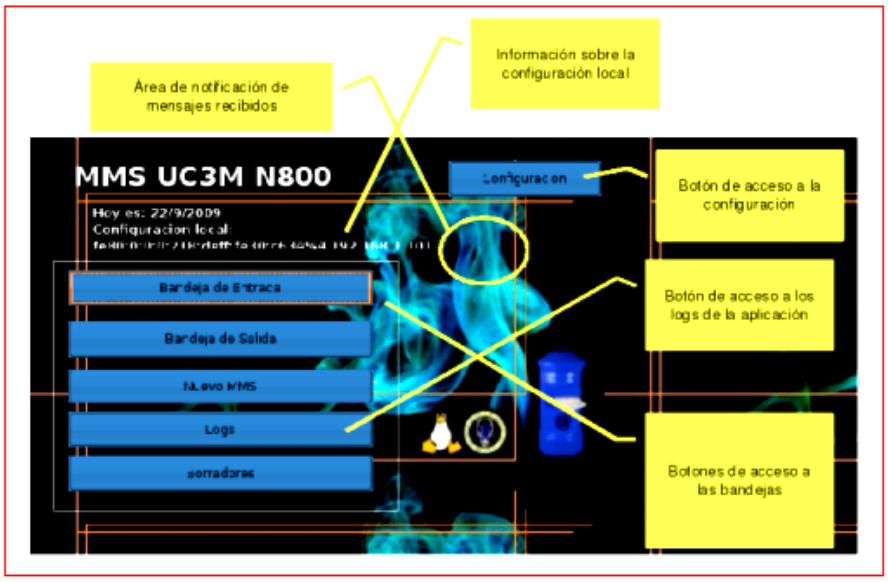


Ilustración 20 - Zonas pantalla principal

3.4.3. BANDEJAS

La estructura de las distintas bandejas de la aplicación, tres en total (entrada, salida y borradores), será la misma.

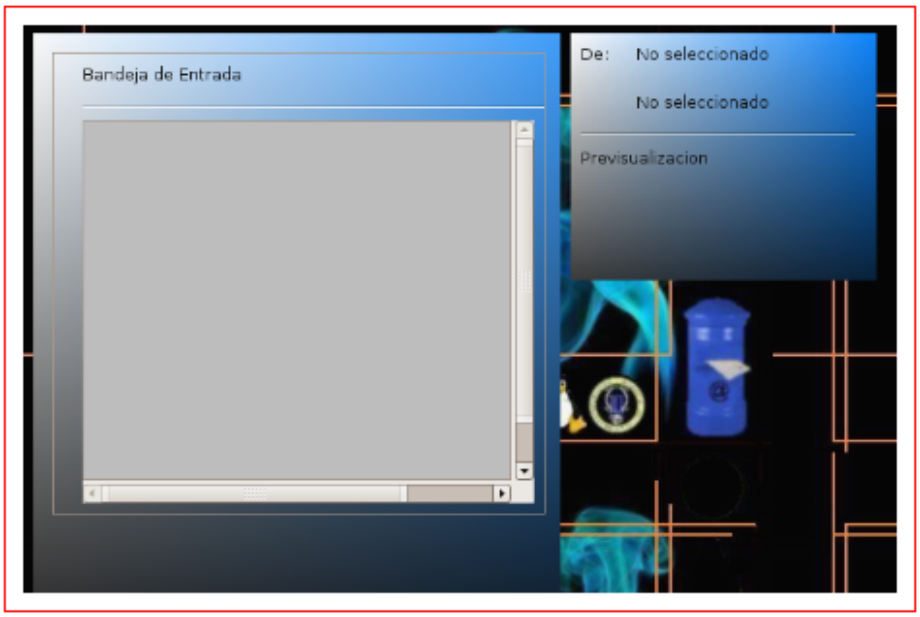


Ilustración 21 - Bandeja de entrada

Constará de dos contenedores fundamentales. El contenedor principal almacenará los mensajes que albergue la bandeja. Cuando se seleccione uno de los mensajes almacenados en el contenedor principal, en el contenedor secundario se mostrará la información del mensaje tal y como se muestra en la siguiente figura.

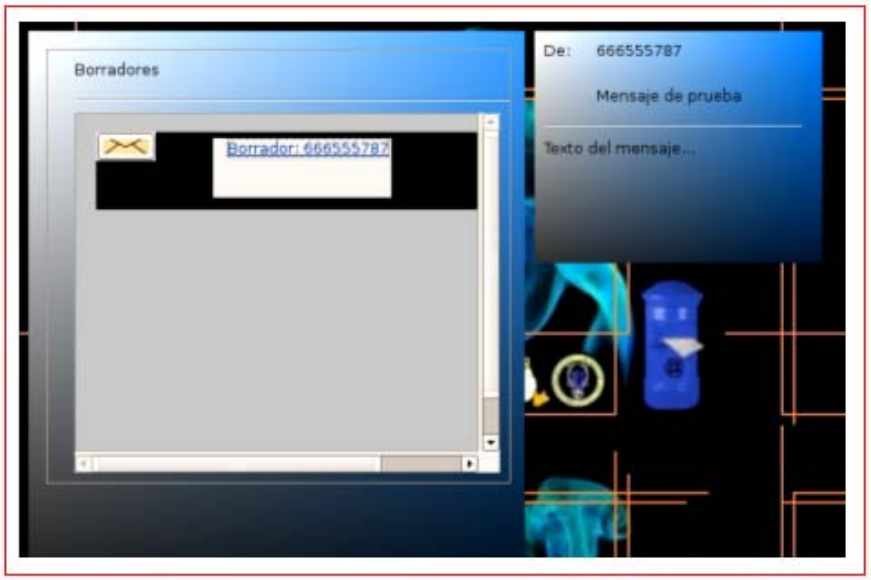


Ilustración 22 - Bandeja de Borradores-Mensaje no leído

Los mensajes en el contenedor podrán mostrar un sobre amarillo si el mensaje no ha sido leído o un sobre gris si el mensaje ha sido leído.

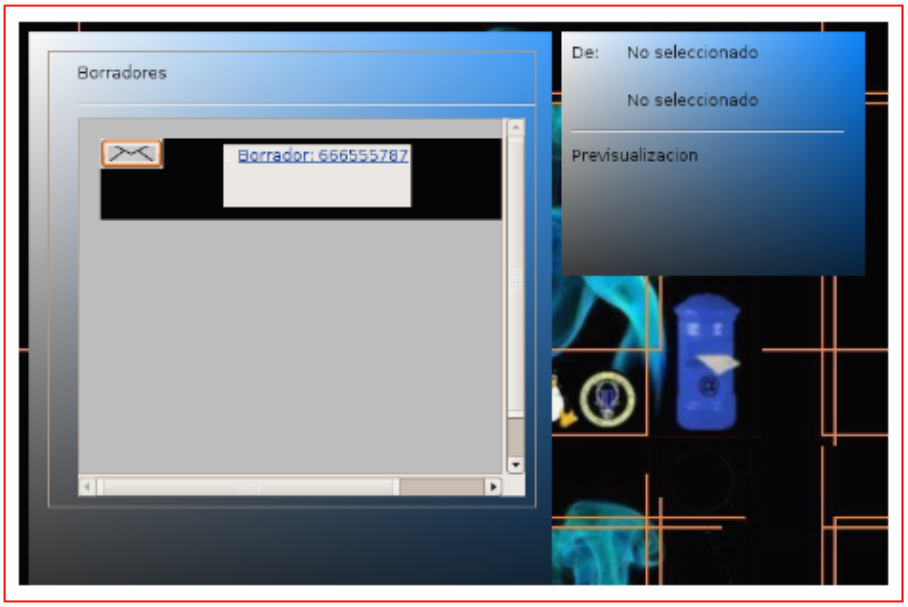


Ilustración 23 - Bandeja de borradores-Mensaje leído

Para abrir los mensajes bastará con pulsar en el sobre que se encuentra en la esquina superior izquierda del subcontenedor que alberga el mensaje.

3.4.4. NUEVO MENSAJE

La pantalla de nuevo mensaje contará con los contenedores necesarios para añadir al MMS todas las fuentes multimedia deseadas: imágenes, archivos de audio, archivos de video y texto.

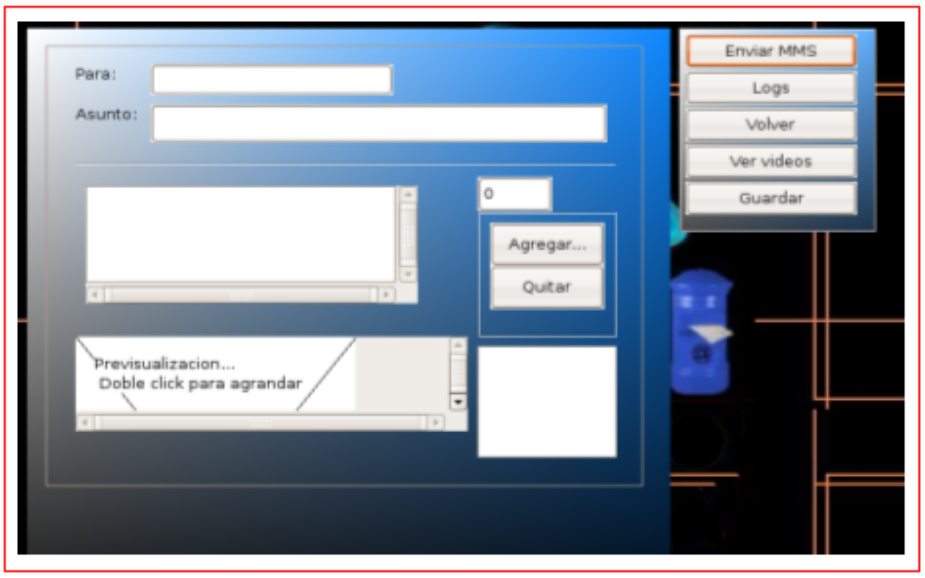


Ilustración 24 - Pantalla nuevo mensaje

Los campos Para y Asunto son los convencionales en sistemas de mensajería tipo mail. Se añade un contenedor para texto y un contador de caracteres. Los archivos multimedia se gestionarán como adjuntos que se almacenarán en el contenedor situado en la posición inferior derecha. Estos archivos podrán añadirse o eliminarse mediante los botones habilitados a tal efecto.

Se añade un campo previsualización para visualizar las imágenes y un botón de previsualización de videos que delegará en la aplicación del terminal para visualizar videos.

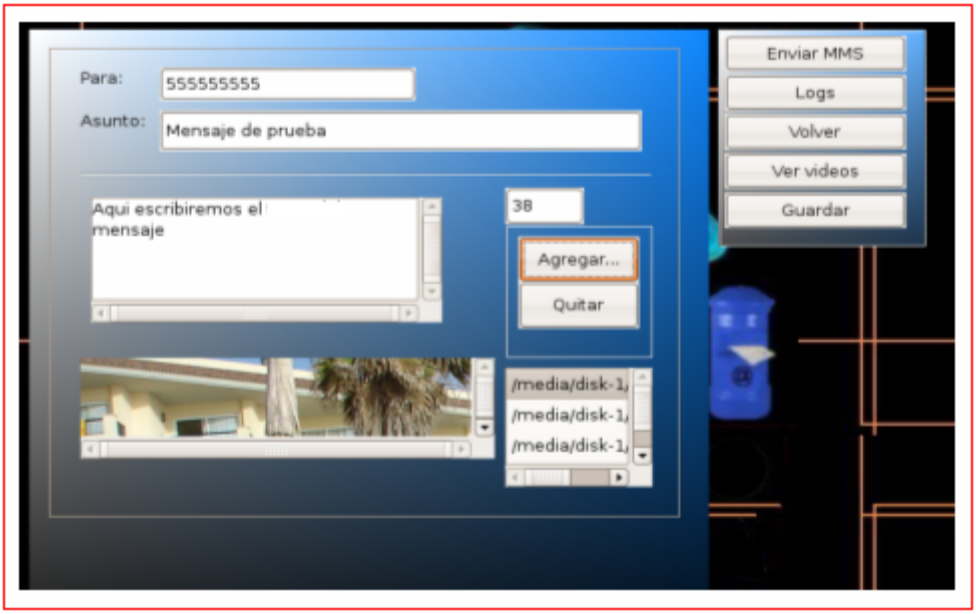


Ilustración 25 - Pantalla nuevo mensaje-rellenado de campos

Se proveerá a la pantalla de un botón para enviar el mensaje así como de otro para guardarlo en la carpeta borradores si no se desea enviar todavía el mensaje.

También, para desarrolladores, se habilita un acceso directo a los logs de la aplicación.

3.4.5. CONFIGURACIÓN

La pantalla de configuración constará de un campo editable para introducir la dirección del servicio de escucha del terminal remoto, sea un centro de mensajes o sea otro terminal. La dirección se compondrá de IP más el puerto de escucha.

Asimismo, se proporcionará otro campo editable para poder variar el puerto de escucha del terminal o servicio cliente.

Ilustración 26 - Pantalla de configuración

3.4.6.LOGS

Por las diferentes pantallas de la aplicación, así como en la interfaz de entrada, habrá un botón de acceso directo a los logs de la aplicación, utilidad implementada para desarrolladores. La pantalla será un simple contenedor de texto no editable.

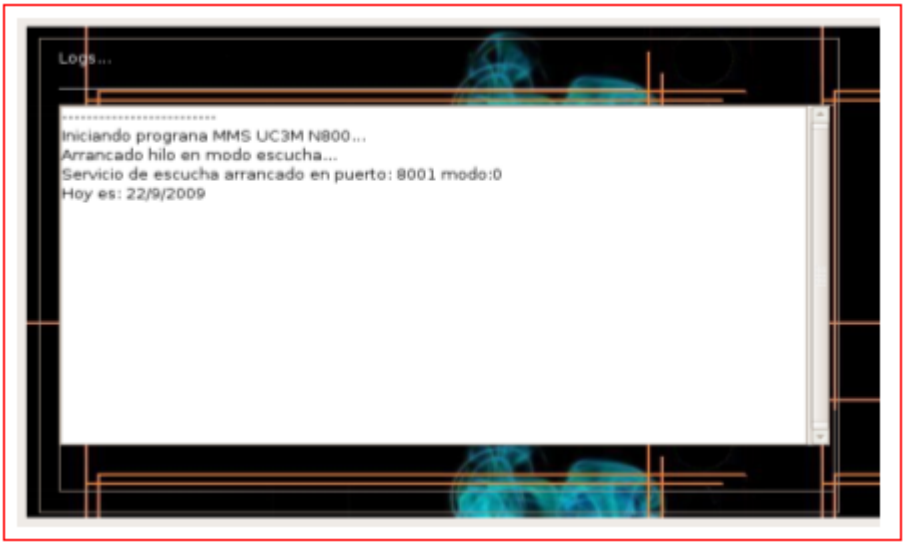


Ilustración 27 - Pantalla de logs

4. ANÁLISIS TÉCNICO

4.1. CARACTERÍSTICAS GENERALES

4.1.1. INFRAESTRUCTURA – TERMINAL

4.1.1.1. Terminal N800 Características

El dispositivo utilizado para el proyecto es la Internet Tablet PC Nokia N800 cuyas características son:

Ilustración 28 - Terminal Nokia 800

Características N800.
❖ Volumen 138 cc ❖ Peso (máx.) 206 g ❖ Largo (máx.) 144mm ❖ Ancho (máx.) 75mm ❖ Espesura 13 (/18mm)
❖ Pantalla táctil de alta resolución (800 x 480 pixels) con hasta 65,536 colores
❖ DDR RAM 128MB ❖ Flash 256MB, 128 MiniSD ❖ Dos ranuras internas para memory card, compatible con SD, MicroSD, MiniSD, MMC, and RS-MMC. Soporta tarjetas de memoria hasta 8GB. SD cards por encima de los 2GB deben ser SDHC compatibles.
❖ Teclado táctil en pantalla y a pantalla completa ❖ Reconocimiento de escritura
❖ Audio: AAC, AMR, AWB, M4A, MP2, MP3, RA (RealAudio), WAV, WMA ❖ Imagen: BMP, GIF, ICO, JPEG, PNG, TIFF, SVG-tiny ❖ Video: 3GP, AVI, H.263, MPEG-1, MPEG-4, RV (Real Video) ❖ Internet radio playlists: M3U, PLS
❖ WLAN: 802.11b/g ❖ Bluetooth: 2.0. ❖ Nokia AV conector 3.5mm ❖ Perfiles soportados: Dial-up Networking, File Transfer, Generic Access, SIM Access, Object Push Profile, Human Interface Profile, and Serial Port profiles ❖ USB 2.0 high speed device mode for PC connectivity
❖ Batería (BP-5L) o En uso:3,5 h o En espera: 13 días
❖ Acceso a Web o Display de alta resolución y ángulo de visión mejorado. o Opera 8 o Adobe® Flash® 9 browser plug-in ❖ Internet

- Soporte para Skype
- Webcam integrada
- Instant messaging
- Email Cliente Multi-protocolo
- Teclado táctil a pantalla completa

❖ Acceso a internet
- Altavoces estéreo de alta calidad
- Media player
- UPnP arquitectura
- Real Rhapsody
- Memoria Expandible

❖ Internet Tablet OS 2007
❖ Aplicaciones clave
- Navegador con Adobe® Flash® 9 plug-in
- Internet calling with video
- Instant Messaging
- Email
- Media player
- Internet Radio
- RSS Feed Reader

❖ Conectividad
- Conectividad automática a través de hotspots o teléfono con Bluetooth compatible.

❖ Display functions
- Zooming, full-screen.

❖ Utilidades
- Gestor de aplicaciones
- Visor de documentos PDF
- Cuaderno de bocetos
- Gestor de ficheros
- Sistema de copia de seguridad
- Visor de imágenes
- Notas
- Calculadora

Tabla 17 - Especificaciones N800

Para la realización de este proyecto resultan interesantes especialmente las siguientes características: el sistema operativo para el desarrollo de la solución, la intefaz inalámbrica para la comunicación, la pantalla táctil y el reproductor de video embebido.

4.2. Solución propuesta

4.2.1. SWT SOBRE JALIMO EN MAEMO CHINOOK

Atendiendo a los requisitos de usuario así como a los requisitos deseados, la solución propuesta debe ser eminentemente multiplataforma. La aplicación debe poder ejecutarse en diferentes entornos y sistemas operativos, así como tener una interfaz gráfica sencilla e intuitiva.

Debido a la naturaleza del proyecto la aplicación debe ser autónoma y portable. Cuando decimos autónoma nos referimos a aplicaciones *Standalone*, es decir, que la aplicación se instala y ejecuta directamente en el dispositivo.

Cuando se habla de multiplataforma inmediatamente pensamos en el lenguaje de programación con más auge en los últimos años tanto en entornos educativos como empresariales, JAVA. Pero para poder implementar la aplicación en este lenguaje de programación, el entorno debe permitir que una máquina virtual corra sobre él.

El entorno que viene de serie en la N800 (Bora 2007) no tiene soporte para la plataforma Jalimo que provee la JVM Cacao sobre la que podría correr nuestra aplicación por lo que necesitaremos hacer un *upgrade* del SO para pasar a la versión Chinook que ya soporta la plataforma Jalimo.

Una vez actualizado el sistema operativo podremos ejecutar una aplicación Java. Puesto que la aplicación debe proveer una interfaz gráfica y ser standalone, se opta por la tecnología SWT. Además optando por esta tecnología, nuestra aplicación podrá correr sobre cualquier sistema operativo (sólo importando en el proyecto la librería SWT adecuada para el SO) con la particularidad de que su aspecto variará dependiendo del SO, ya que las interfaces programadas con SWT aprovechan las interfaces gráficas nativas del sistema operativo.

Así pues, para la solución específica propuesta en el proyecto, la aplicación se desarrollará en java con SWT (Widgets) y se ejecutará sobre la Cacao JVM (plataforma Jalimo) en el sistema Maemo Chinook para el terminal N800.

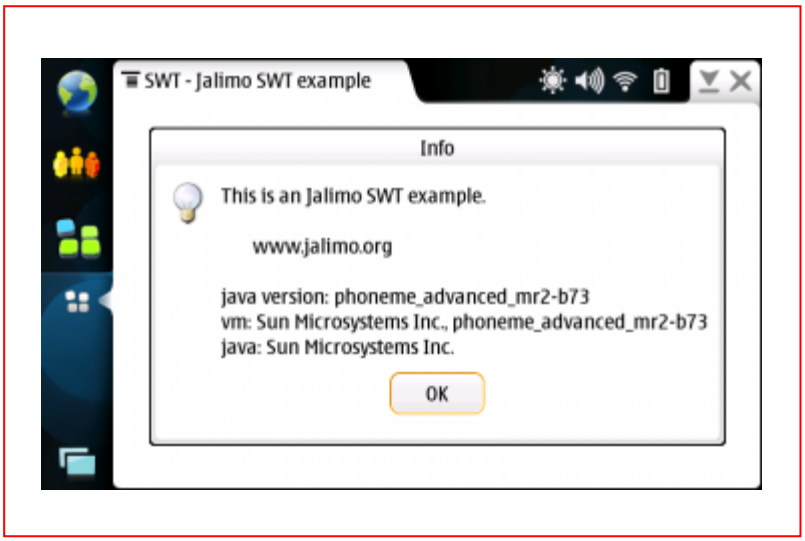


Ilustración 29 - SWT sobre Jalimo en Chinook obtenida de la página de Jalimo.

4.2.2. DESCRIPCIÓN DE LA SOLUCIÓN

El programa puede ser descrito en los siguientes términos:

- Lenguaje de programación: **Java**
- Interfaz gráfica: **SWT**
- Gestión MMS: Para la composición, el envío y recepción de mensajes multimedia se utilizarán la librería Nokia **MMSLibrary**.jar.
- Gestión de los recusos: El programa será **multihilo**, siendo el hilo principal el encargado de gestionar la intefaz gráfica y las acciones básicas de usuario, y el secundario el que se encargue de la gestión de los MMS.
- Reproductores: Dado que el sistema de MMS debe permitir la visualización de video y escuchar los archivos de sonido, optaremos por dos estrategias diferentes. En cuanto a los archivos de sonido, utilizaremos la **MP3 library para plataformas java**. Para los archivos de video utilizaremos el reproductor de video embebido en el terminal o dispositivo.

La aplicación, desarrollada con SWT, dependiendo del sistema operativo, correrá sobre los entornos Java de ejecución Jalimo o JRE estándar. Las máquinas virtuales en los entornos que utilicen Jalimo, serán Cacao para Maemo y phoneME para Windows Mobile.

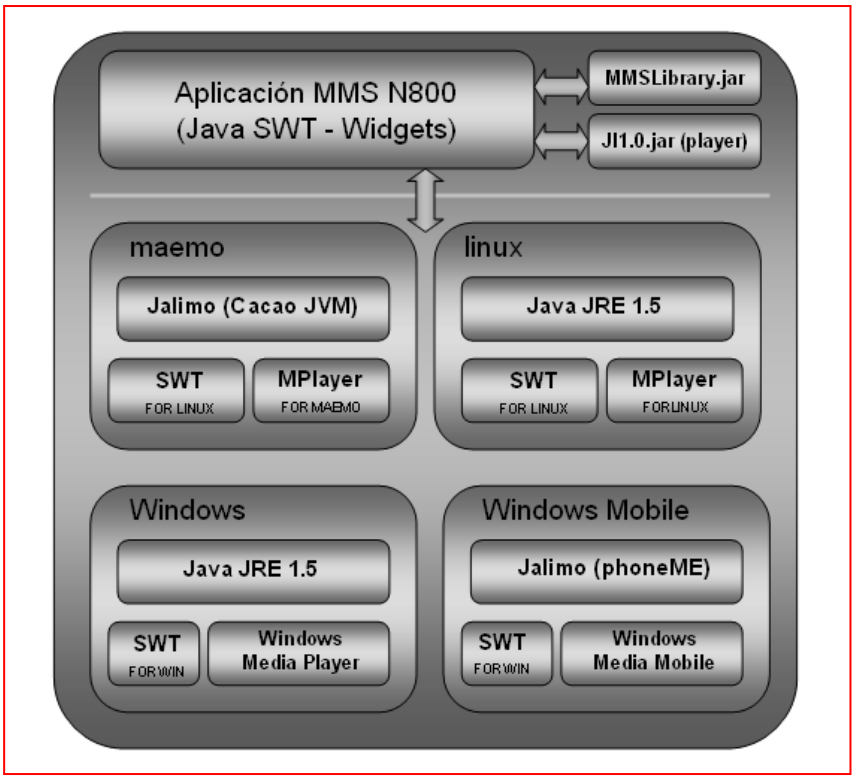


Ilustración 30 - Aquitectura software solución propuesta

4.2.3. DESCRIPCIÓN DE ELEMENTOS

4.2.3.1. Java

Las características principales de este lenguaje de programación [23] se describen a continuación.

Ilustración 31 - Logo de Java

- **Orientado a objetos**

Aunque tiene influencias de sus predecesores, Java no fue diseñado para tener un código fuente compatible con otros lenguajes. Esto proporcionó la libertad necesaria para diseñar un lenguaje partiendo de cero. El resultado fue una aproximación limpia, útil y pragmática a los objetos. Java tomó prestadas ideas de muchos entornos y seminarios de orientación a objetos de hace algunas décadas, consiguiendo un equilibrio entre el modelo purista (“cualquier cosa es un objeto”) y el modelo pragmático (“no necesito trabajar con objetos”). El modelo de objeto que define Java es simple y fácil de ampliar, mientras que los tipos simples, como los enteros, permanecen como “no objetos” de alto rendimiento.

- **Distribuido**

Java fue diseñado para el entorno distribuido de Internet, por lo que trabaja con el protocolo TCP/IP. De hecho, el acceso a un recurso utilizando una URL no es muy diferente a acceder a un archivo. La versión original de Java (llamada Oak) incluía facilidades para que los objetos pudieran ejecutar procedimientos de forma remota. Java dispone de estas interfaces en un paquete de invocación de método remoto (RMI). Esta característica le aporta un nuevo nivel de abstracción en la programación cliente / servidor.

- **Arquitectura neutral**

Un aspecto importante para los diseñadores de Java era la longevidad y portabilidad del código. Uno de los principales problemas con el que se enfrentan los programadores es que no hay garantías de que un programa que escriben hoy se ejecutará mañana, incluso en la misma máquina. Las actualizaciones del sistema operativo, de los procesadores y los cambios en los recursos básicos del sistema pueden provocar que un programa deje de funcionar correctamente. Los diseñadores de Java tomaron varias decisiones difíciles en el lenguaje y en el intérprete para intentar cambiar esta situación. Su principal objetivo era escribir una vez, ejecutar en cualquier sitio ("Write Once, Run Everywhere").

- **Portable**

Hoy en día se están utilizando muchos tipos de computadoras y de sistemas operativos en todo el mundo, y muchas de estas computadoras están conectadas a Internet. Por tanto, es necesario tener algún mecanismo que permita generar código ejecutable que sea portable a todos esos tipos de plataformas que están conectadas a Internet. El mismo mecanismo que ayuda a garantizar la seguridad es que también ayuda a crear la portabilidad. De hecho, la solución a estos dos problemas es elegante y eficiente.

- **Interpretado**

Java permite la creación de programas que se pueden ejecutar en varias plataformas compilando en una representación intermedia llamada código binario (Java bytecode). Este código se puede interpretar en cualquier sistema que tenga un intérprete de Java. La mayoría de los intentos anteriores que se habían realizado para aportar una solución independiente de la plataforma no tenía un rendimiento muy bueno. Otros sistemas interpretados sufren un déficit de rendimiento casi insalvable. Sin embargo, Java fue diseñado con cuidado de manera que fuese sencillo traducirlo directamente a código máquina nativo para conseguir un rendimiento alto. Los intérpretes de Java son los que realizan esta optimización y no pierden ninguno de los beneficios del código neutral a la plataforma. La frase "independencia de la plataforma con alto rendimiento" ya no es un contrasentido.

- **Robusto**

El entorno multiplataforma de la red le exige bastante a un programa, ya que tiene que ejecutarse en una gran variedad de sistemas. Por esta razón, la capacidad de crear programas robustos tuvo un prioridad en el diseño de Java. Para ganar en fiabilidad, Java le restringe en unas cuantas áreas para obligarle a encontrar pronto los errores que se producen en el desarrollo de un programa. A la vez, Java le libera de tener que preocuparse por muchas de las principales causas de error en la programación. Como Java es un lenguaje fuertemente tipado, le permite comprobar el código en tiempo de compilación. Sin embargo, también comprueba el código en tiempo de ejecución. De hecho, muchos de esos errores difíciles de encontrar que se convierten a menudo en situaciones en tiempo de ejecución difíciles de reproducir son imposibles de crear en Java. Saber que lo que uno ha escrito se comportará de una manera predecible bajo diversas condiciones es una de las principales características de Java.

Para entender por qué Java es robusto, consideremos las dos principales causas por las que un programa puede fallar: gestión de memoria y condiciones excepcionales no controladas (errores en tiempo de ejecución). La gestión de memoria puede ser una tarea compleja y tediosa, por ejemplo, en C el programador debe manualmente reservar y liberar toda la memoria dinámica. Algunas veces esto provoca problemas. Por ejemplo, si olvidas liberar la memoria que reservaste en un principio. O peor aún, se puede intentar liberar memoria que otra parte del programa todavía está utilizando. Java elimina este problema al gestionar internamente la memoria, tanto al reservarla como al liberarla. De hecho, la liberación se realiza de manera totalmente automática, ya que Java proporciona un recolector de basura (reciclaje de memoria) para los objetos que no se utilizan. Las condiciones excepcionales en los entornos tradicionales surgen a menudo en situaciones como la división por cero o un archivo no encontrado y deben ser gestionadas mediante construcciones torpes y difíciles de leer. Java proporciona una gestión de excepciones orientada a objetos. En un programa Java escrito correctamente, todos los errores de ejecución pueden y deben ser gestionados por el programa.

- **Seguro**

Como es sabido, cada vez que se transfiere a una computadora un programa, ésta corre el peligro de recibir un virus. Antes de aparecer Java, la mayoría de los usuarios no transferían habitualmente programas ejecutables y aquellos que lo hacían los

analizaban para ver si tenían algún virus antes de ejecutarlos. Incluso así, la mayoría de los usuarios estaban preocupados por la posibilidad de infectar sus ordenadores. Además de los virus, hay otros programas maliciosos de los que hay que protegerse. Este tipo de programas pueden recoger información privada, como números de tarjetas de crédito, cuentas bancarias, claves personales, etc., analizando el contenido del sistema de archivos local de la computadora. Java responde a estas dos cuestiones estableciendo un cortafuegos (firewall) entre una aplicación de red y la computadora local.

- **Dinámico**

Los programas Java se transportan con cierta cantidad de información que se utiliza para verificar y resolver los accesos a los objetos en tiempo de ejecución.

4.2.3.1.1. Arquitectura básica

Inicialmente pensadas para correr en dispositivos electrodomésticos como sistemas embebidos, el paradigma de Java [8] fue ganando terreno en áreas más tradicionales de la computación y sistemas de información. Las aplicaciones Java se crean a través de un proceso de compilación en tiempo de diseño, cuyo código objeto no es código máquina de alguna plataforma establecida, sino el correspondiente a una máquina "virtual", entendiendo por maquina virtual (VM por sus iniciales en inglés) como la implementación por software de una máquina (ordenador) que ejecuta programas como si fuese una máquina real. La definición más precisa, sin embargo, es la propuesta por (Popek & Goldberg, 1974), donde se establece que una VM es un "duplicado aislado y eficiente de una maquina real".

Un programa escrito en Java recibiría servicios del entorno en de tiempo de ejecución Java (*Java Runtime Environment*, JRE) a través de la emisión de comandos de los cuales el resultado esperado es devuelto por JRE. Al proveer dichos servicios al programa, el software JRE está actuando como una "máquina virtual", tomando el lugar del sistema operativo o hardware para el cual el programa normalmente debería haber sido escrito.

Esencialmente, este código intermedio – conocido como *bytecode*- se ejecutará en una máquina virtual de procesos, que puede ser conceptualizada como un conjunto de servicios. El código máquina es al microprocesador lo que el código intermedio es a

la máquina virtual. En la práctica, el *bytecode* se traduce en bloques a código máquina, mediante un proceso de interpretación, y últimamente se ha conseguido una significativa ganancia en rendimiento a través de técnicas de compilación en tiempo de ejecución sobre demanda (*Just-In-Time compilers*). Como esta compilación se realiza incrementalmente, a petición, subiendo y procesando los bloques necesarios en la aplicación, podemos decir que la compilación final se realiza en memoria, para obtener el código máquina que realmente se ejecuta.

4.2.3.1.2. Portabilidad y rendimiento

Este esquema brinda la portabilidad entre plataformas, debido a que el *bytecode* tiene una estructura normalizada y unificada, restará implementar solamente en cada plataforma (microprocesador + sistema operativo) una VM correspondiente para lograr la deseada portabilidad y "escribir una vez, desplegar muchas" *write-once, deploy-many*.

Respecto del rendimiento, en las versiones iniciales se tenía un esquema interpretado, donde por su propia naturaleza se sufría de una importante ralentización respecto del código máquina puro. Con el advenimiento de los compiladores en tiempo de ejecución JIT (*Just-In-Time compilers*), la velocidad actualmente es de alrededor del 85% de la alcanzada por el código de máquina puro. Estos guarismos se alcanzan porque la máquina virtual Java JVM ejecuta sofisticados procedimientos de optimización interna, compilación en adelanto, encolamiento de métodos (*method-inlining*) y otra docena de optimizaciones, lo que convierten a esta pieza de software en una muy compleja y sofisticada.

4.2.3.1.3. Administración de memoria de la VM

Desde el punto de vista del sistema operativo moderno, la JVM es un proceso nativo más. Pero dentro del proceso de la JVM, existe una gran cantidad de objetos, subprocesos y servicios que se encargan de administrar automáticamente la memoria de la JVM, de cargar y descargar las librerías, de comunicarse entre procesos y hacia otros objetos del sistema operativo, etc. Una característica importante es su capacidad de manejo automático de memoria. En un lenguaje tradicional como el C++, el programador puede incluso escribir a segmentos de memoria que no le corresponden, produciendo una excepción de violación de segmento, o bien "olvidar" el recuperar la memoria de los objetos en desuso, cosa que debe hacer explícitamente. En Java, el

programador sólo debe preocuparse por crear los objetos, al liberarlos es el propio *runtime* quien se encargará de recuperar la memoria. Igualmente el administrador de memoria no permite que un objeto pueda escribir en segmentos de memoria que no le corresponden.

Uno de los servicios esenciales de la máquina virtual se conoce como **recolector de basura** (GC por sus siglas en inglés de *garbage collector*), proceso que se encarga de reclamar y recuperar recursos de memoria en objetos que ya no se utilizan. Mantiene un continuo monitoreo del grafo de objetos en la memoria de la VM, donde recorre este grafo en forma cíclica e indefinida, en cada recorrida se registran los objetos de-referenciados cuya memoria podría ser reclamada. En cada lazo de recorrida y detección, denominado ciclo de recolección, el recolector de basura registra los objetos candidatos a ser eliminados físicamente. En principio, los objetos de-referenciados carecen de punteros, pero su lugar físico queda existente en la memoria de la VM. Para recuperar memoria, el recolector decide cuando hacerlo en función de parámetros que lee respecto de la totalidad de funcionamiento de la máquina virtual. Con lo cual, se acepta que el programador nunca podrá conocer cuando el GC lanzará el proceso de recolección porque en principio no puede conocer los dichos parámetros internos.

Cuando se recupera memoria de objetos de-referenciados, se debe compactar la misma, para lograr que la memoria sin uso quede en un segmento contiguo. Este es un proceso altamente costoso, por lo cual el recolector decide heurísticamente cuando lanzarlo.

Lo descrito hasta ahora se aplica a las máquinas virtuales de escritorio, es decir, a las versiones completas de Java que corren en ordenadores personales, Java SE y EE, diseñadas para gama intermedia y servidores. En los dispositivos móviles, la máquina virtual es normalmente un subconjunto de su contraparte de escritorio, esto debido principalmente a las restricciones de potencia de cómputo, memoria y dispositivos E/S en dichos dispositivos.

4.2.3.1.4. Recolección de basura y finalizadores

Si bien el programador puede forzar la recolección de basura, esto obliga a la máquina virtual a alterar el plan de recolección, penalizando siempre esta acción a

través de una perceptible ralentización de la ejecución. Otro aspecto que se debe evitar es el uso de finalizador, concepto opuesto al de un constructor. Esencialmente, es lo que se ejecuta solamente cuando el objeto es "elegible" para ser recolectado. El recolector marca todos los objetos de-referenciados antes que sean recolectados, y ejecuta el método `finalize` para aquellos objetos que tengan un `finalizer`. El recolector no puede "declinar" el ejecutarlo porque está explícitamente definido en el código del programa. Por lo tanto, en la corrida del recolector, éste "recolecta" todos los objetos "con finalizador" dentro de una cola y acto seguido vacía esta cola invocando el finalizador para cada objeto. Para evitar que estos procesos de terminación de objetos interfieran con el normal proceso de la aplicación, la cola de finalización se ejecuta en otro hilo o *thread* de la máquina virtual distinto del principal, lo que presupone una arquitectura de ejecución "multihilos" en las VM modernas.

El proceso de recolectar, poner en cola e invocar al finalizador de todos estos objetos toma tiempo, pero la real penalización está en el finalizador en sí mismo: no existe forma de conocer que hará el método finalizar, o cuánto tiempo llevará. Esto es justamente lo opuesto a lo que se pretende si la intención del programador es controlar los valiosos recursos del sistema.

Por lo expuesto, la recolección de basura y los finalizadores deben usarse con extrema cautela en escenarios restringidos de recursos, especialmente crítico en sistemas móviles.

4.2.3.1.5. Versiones actuales de Java

Como se ha indicado anteriormente, Java es una plataforma tecnológica, la cual está compuesta de diversas versiones, cada una de ellas pensada para atacar un escenario de sistemas de información definido:

- *Plataforma Java Edición Estándar*: Java SE permite desarrollar y distribuir aplicaciones Java en máquinas de escritorio y servidores, embebidas y en tiempo real. Incluye clases que dan soportan el desarrollo de servicios web basados en Java, y es el bloque fundamental para la Plataforma Java, Edición Empresarial.

- *Plataforma Java Edición Empresarial*: Java EE es un súper-conjunto de la anterior, construida sobre Java SE, es un estándar de la industria para implementar

arquitecturas orientadas a servicios de clase empresarial y aplicaciones web de próxima generación.

- *Plataforma Java Edición Micro*: Java ME es una colección de tecnologías y especificaciones para crear una plataforma que se ajuste a los requerimientos de dispositivos móviles tales como productos de consumo, dispositivos embebidos y dispositivos móviles avanzados. Estas tecnologías y especificaciones se combinan para crear un entorno de ejecución implementado completamente en Java, ajustado para satisfacer un determinado dispositivo o mercado.

4.2.3.1.6. Extendiendo la plataforma

Para proponer nuevas tecnologías en Java, se creó en 1998 un proceso denominado Proceso de la Comunidad Java (JCP por sus iniciales en inglés), que consiste en permitir que asociados de negocios se involucren a través de un mecanismo formal en la definición de futuras versiones de la plataforma Java. Los documentos de trabajo de JCP se denominan Pedidos de Especificación de Java, (JSR por sus iniciales en inglés), y allí se describen las especificaciones propuestas y las tecnologías que serían agregadas a la plataforma Java. Se realizan revisiones formales de estos documentos hasta que se obtiene una versión definitiva, la cual se somete a votación por un comité ejecutivo del proceso JCP. El documento JSR así aprobado se transforma en una *referencia de implementación*, en la forma de una implementación gratuita de dicha tecnología en código fuente, más un *kit* de compatibilidad de tecnología, que se utiliza para verificar la especificación de la nueva API.

4.2.3.2. Jalimo

El proyecto Jalimo [11] intenta crear un entorno de ejecución libre para aplicaciones java en dispositivos móviles con Linux con SO. Se utiliza en varias plataformas de desarrollo como OpenMoko o Maemo.

Jalimo consiste en una colección de paquetes de software *open source* que implementan diferentes partes del entorno en tiempo de ejecución de java así como otros productos relacionados. Estos productos se empaquetan en una distribución común hecha a medida para trabajar con las plataformas objetivo.

Jalimo contiene dos máquinas virtuales Cacao JVM y JamVM, de las cuales la primera usa la compilación *Just in time* y la segunda está basada en interpretación.

También contiene la librería GNU ClassPath, que junto con la JVM forma lo que se conoce como *Java Runtime Environment* o entorno en tiempo de ejecución de Java. El proyecto también incluye paquetes que incorporan componentes de interfaz gráfica de usuario, así como otras funcionalidades.

4.2.3.3. Cacao

La Cacao JVM [4] es una máquina virtual Java que usa compilación *Just in Time*, lo que significa que el bytecode es compilado en código-máquina específico la primera vez que se ejecuta.

Antes, las máquinas virtuales usaban interpretación, lo que significa que cada vez que cada vez que se ejecutaba una línea de bytecode, éste era evaluado y ejecutado, sin reutilizar la traducción a código máquina de otras ejecuciones.

Cacao fue creado en 1997 como un proyecto de investigación, pero en 2004 el proyecto fue liberado bajo una licencia GPL y ahora se desarrolla como proyecto de código libre.

Está disponible para la arquitectura de procesador ARM, y par alas arquitecturas Intel x86 lo que hace posible que también pueda correr en PC's. Además de hacerlo en Linux, también puede correr en otros sistemas operativos tales como FreeBSD y Mac OS X.

4.2.3.4. GNU ClassPath

GNU ClassPath [9] es un proyecto que intenta crear una implementación libre de la biblioteca de clases estándar Java que se usa en muchas implementaciones de JVM. Por ejemplo, en ambas máquinas virtuales del proyecto Jalimo, Cacao JVM y JamVM, se usa GNU ClassPath como núcleo de biblioteca de clases Java.

4.2.3.5. phoneME

El proyecto phoneME [27] es la implementación de Sun Microsystems de la máquina virtual de Java y las librerías asociadas de Java ME, licencia bajo GNU General Public License.

La biblioteca de phoneME incluye las implementaciones de Connected Limited Device Configuration (CLDC) y Mobile Information Device Profile (MIDP) así como una implementación completa o parcial de algunos paquetes opcionales JSRs.

4.2.3.6. SWT

SWT [30] (siglas en inglés de Standard Widget Toolkit) es un conjunto de componentes para construir interfaces gráficas en Java, (widgets) desarrollados por el proyecto Eclipse. Recupera la idea original de la biblioteca AWT de utilizar componentes nativos, con lo que adopta un estilo más consistente en todas las plataformas, pero evita caer en las limitaciones de ésta. La biblioteca Swing, por otro lado, está codificada enteramente en Java y frecuentemente se la acusa de no brindar una experiencia idéntica a la de una aplicación nativa. Sin embargo, el precio a pagar por esa mejora es la dependencia (a nivel de aspecto visual y no de interfaz de programación) de la aplicación resultante del sistema operativo sobre el cual se ejecuta. La interfaz del workbench de eclipse también depende de una capa intermedia de interfaz gráfica de usuario (GUI) llamada JFace que simplifica la construcción de aplicaciones basadas en SWT.

Este framework, que crea a través de (Java Native Interface) interfaces gráficas nativas del Sistema Operativo en donde ejecutemos nuestra aplicación SWT, lo que implica es que con el mismo código visualizaremos en cada Sistema Operativo nuestras ventanas como si hubieran sido creadas para ese SO en específico.

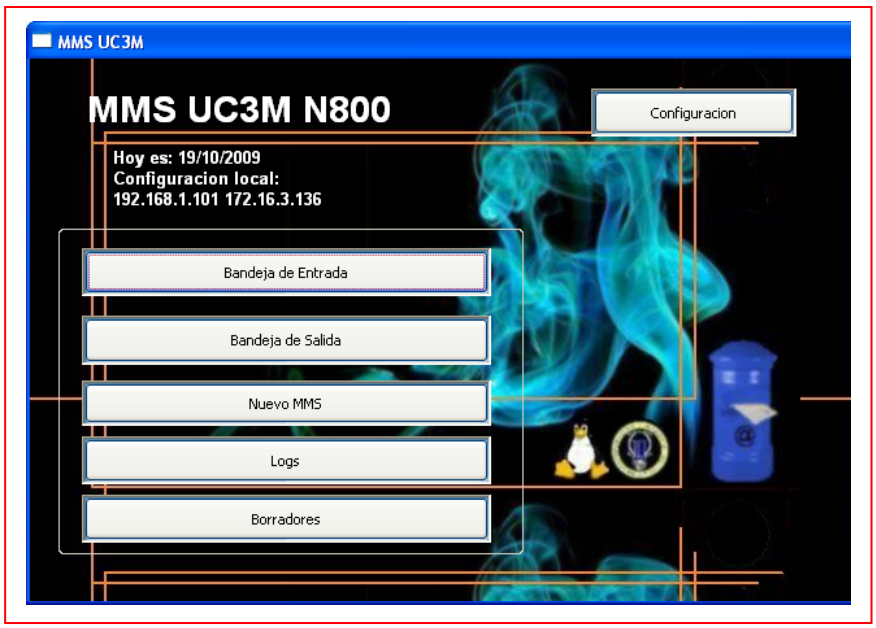


Ilustración 32 - Aplicación ejecutada en Windows

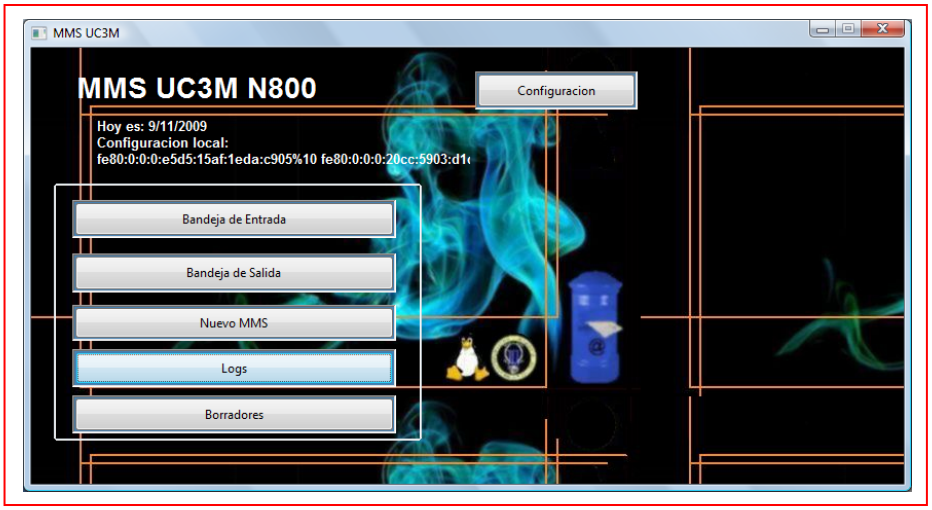


Ilustración 33 - Aplicación ejecutada en Windows Vista

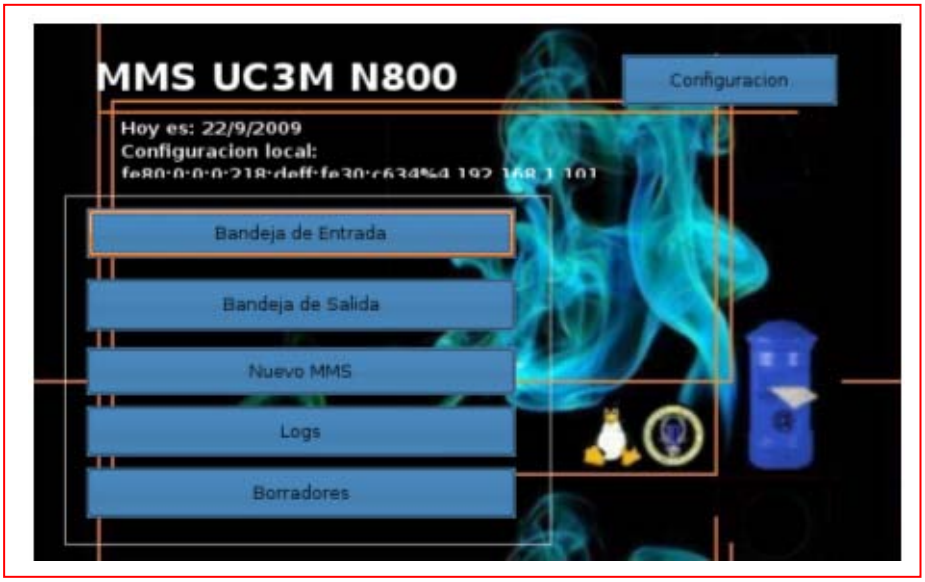


Ilustración 34 - Aplicación ejecutada en Linux

Ilustración 35 - Aplicación ejecutada en Maemo-N800.

Ventajas:

- Ya que crea nativamente la GUI dependiendo del SO, es más rápido que Swing
- Consume mucho menos
- La interfaz gráfica se verá como las demás aplicaciones del SO
- Está en constante desarrollo

Ilustración 36 - Pagina oficial SWT (http://www.eclipse.org/swt/)

4.2.3.6.1. Widget

En el contexto de la programación de aplicaciones visuales [30], los widgets tienen un significado más amplio como componente o control visual que el programador reutiliza y tienen un gran valor para el usuario, idea íntimamente ligada con el concepto de interfaz gráfica de usuario (GUI por sus siglas en inglés).

Desde esta perspectiva, un widget, también conocido como artilugio o control, es un componente gráfico, o control, con el cual el usuario interactúa, como por ejemplo, una ventana, una barra de tareas o una caja de texto.

4.2.3.7. MP3 Library para Java

La librería JLayer 1.0 provee las funcionalidades necesarias para decodificar, convertir y reproducir MP3 en plataformas JAVA. En nuestro proyecto utilizaremos la Release 1.0 (Licencia LGPL).

4.2.3.8. Log4j

Log4j [13] es una biblioteca open source desarrollada en Java por la Apache Software Foundation, que permite a los desarrolladores de software elegir la salida y el

nivel de granularidad de los mensajes o “logs” (logging) en tiempo de ejecución y no en tiempo de compilación como es comúnmente realizado. La configuración de salida y granularidad de los mensajes es realizada en tiempo de ejecución mediante el uso de archivos de configuración externos.

Por defecto Log4J tiene 6 niveles de prioridad para los mensajes (trace, debug, info, warn, error, fatal). Además existen otros dos niveles extras (all y off):

Niveles de prioridad (De mayor -poco detalle- a menor -mucho detalle-):

- FATAL: se utiliza para mensajes críticos del sistema, generalmente después de guardar el mensaje el programa abortará.
- ERROR: se utiliza en mensajes de error de la aplicación que se desea guardar, estos eventos afectan al programa pero le dejan seguir funcionando, como por ejemplo que algún parámetro de configuración no es correcto y se carga el parámetro por defecto.
- WARN: se utiliza para mensajes de alerta sobre eventos de los que se desea mantener constancia, pero que no afectan al correcto funcionamiento del programa.
- INFO: se utiliza para mensajes similares al modo "verbose" en otras aplicaciones.
- DEBUG: se utiliza para escribir mensajes de depuración. Este nivel no debe estar activado cuando la aplicación se encuentre en producción.
- TRACE: se utiliza para mostrar mensajes con un mayor nivel de detalle que debug.

Extras:

- ALL: este es el nivel de máximo detalle, habilita todos los logs (en general equivale a TRACE).
- OFF: este es el nivel de mínimo detalle, deshabilita todos los logs.

La aplicación lleva activada por defecto la opción con más detalle, pero el desarrollador o usuario de la aplicación puede modificar el nivel de los logs modificando simplemente el fichero de configuración log4j.properties.

4.3. DISEÑO SOFTWARE

4.3.1. DIAGRAMA DE CLASES

A partir de los casos de uso identificados para el sistema se realizó el diagrama de clases del dominio, que se corresponde con las siguientes figuras:

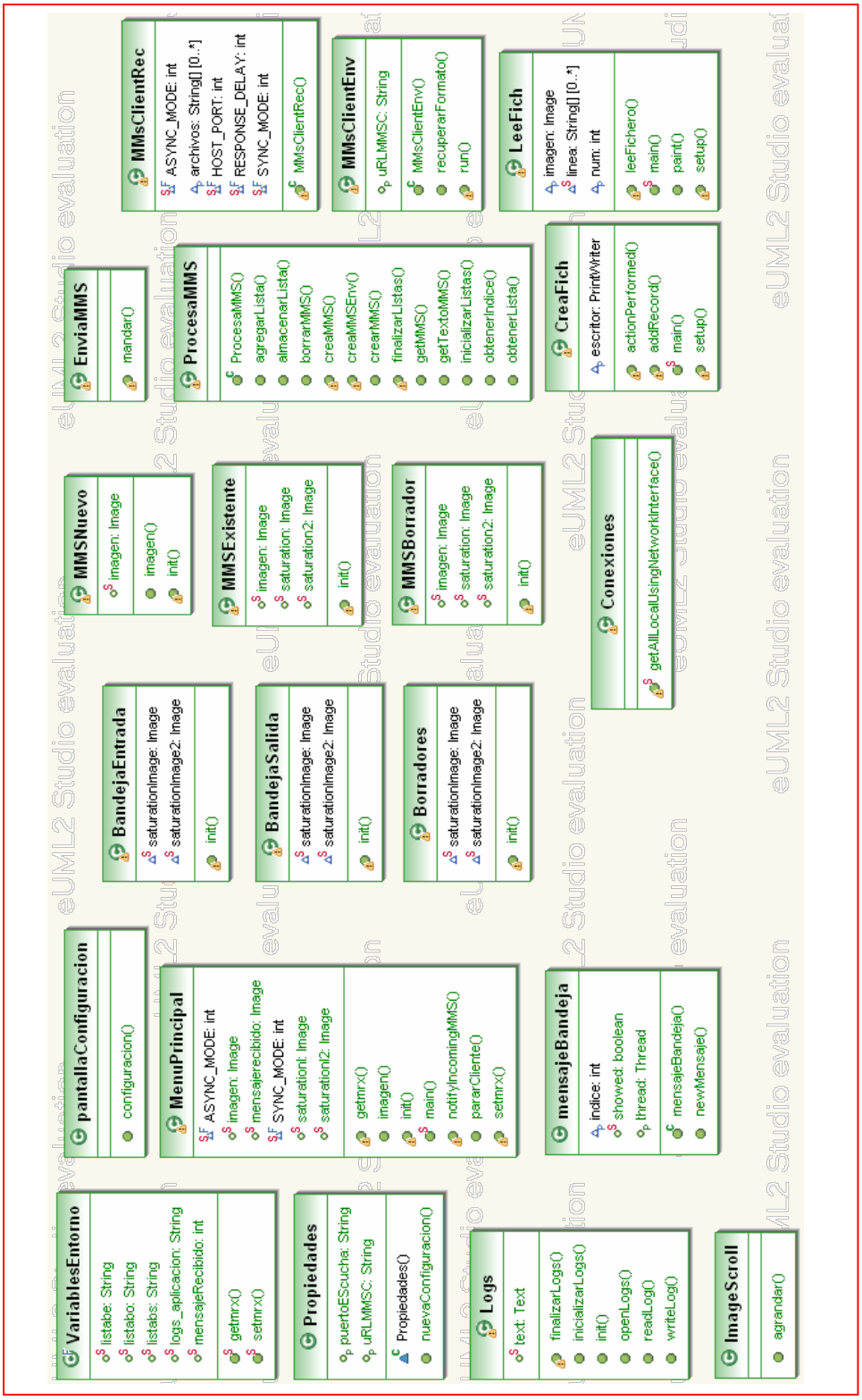


Ilustración 37 - Representación clases Proyecto N800MMS

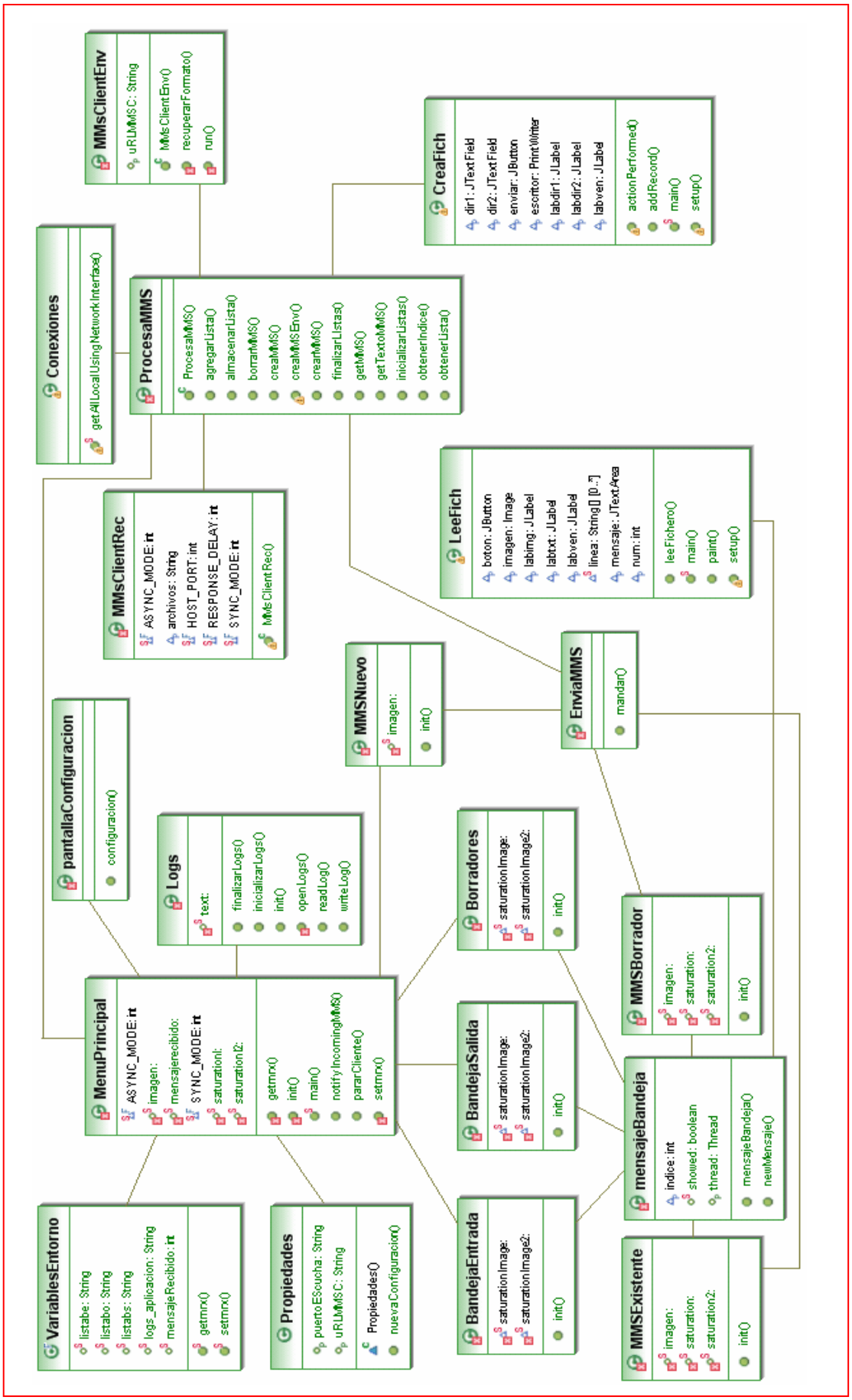


Ilustración 38 - Diagrama de clases y dependencias Proyecto N800MMS

4.3.2. DIAGRAMA DE ESTADOS

Cuando arranquemos el programa, para optimizar el rendimiento del mismo una vez en funcionamiento, realizaremos una carga inicial de variables de entorno. También cargaremos los ficheros que contienen la información de indexación de los MMS en bandeja que tiene el usuario. La información genérica del entorno de ejecución se mantendrá en variables globales durante el tiempo en que el programa esté arrancado.

Dado que el programa debe atender las peticiones del usuario y al servicio de mensajería de forma continua y paralela, al arrancar el programa arrancaremos dos hilos de ejecución. Uno de los hilos se encargará de la interfaz gráfica y de atender las acciones del usuario. El otro se encargará de mantener una escucha en segundo plano por si se reciben MMS y se comunicará con la interfaz gráfica para mostrar una alerta al usuario de mensaje recibido.

Al final de la ejecución del programa descargaremos la información de la ejecución en el punto de finalización en los ficheros de datos y cerraremos el programa.

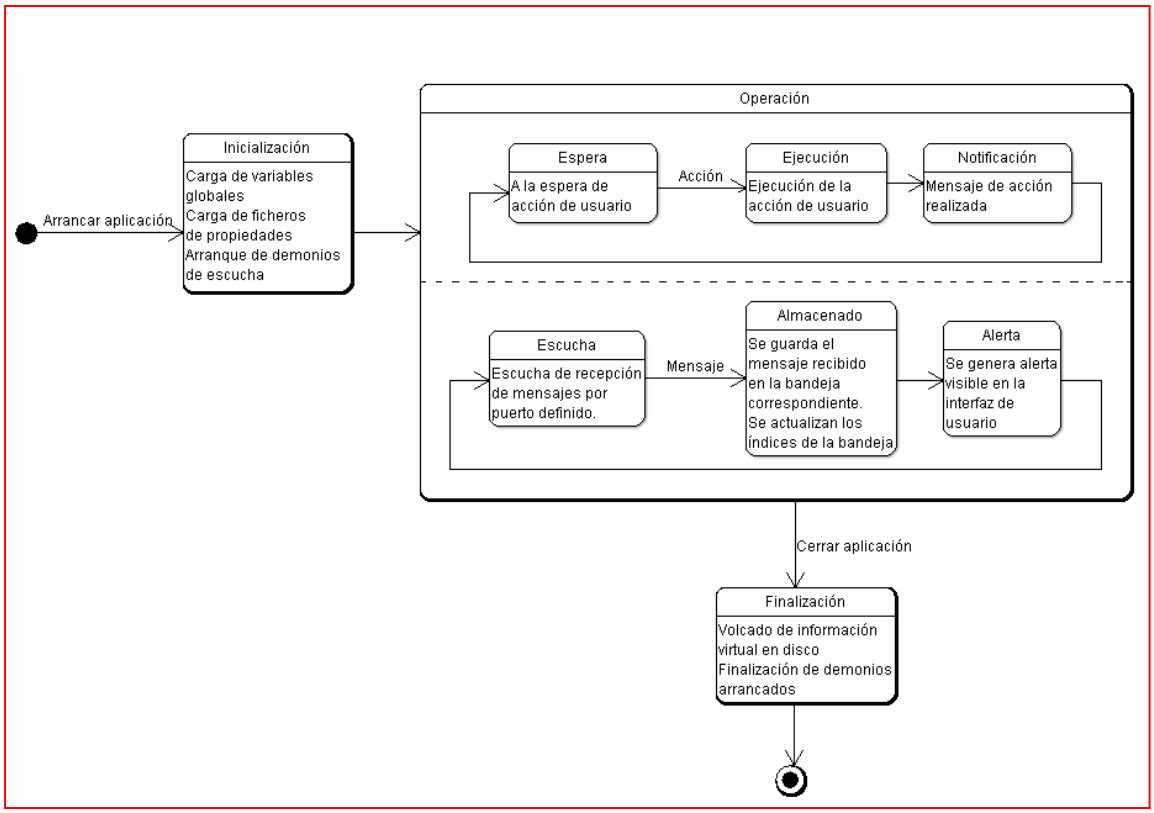


Ilustración 39 - Diagrama de estados del proyecto N800MMS

4.3.3. DIAGRAMA DE SECUENCIA

El diagrama de secuencia de las interacciones más complejas de la aplicación puede resumirse en la siguiente figura:

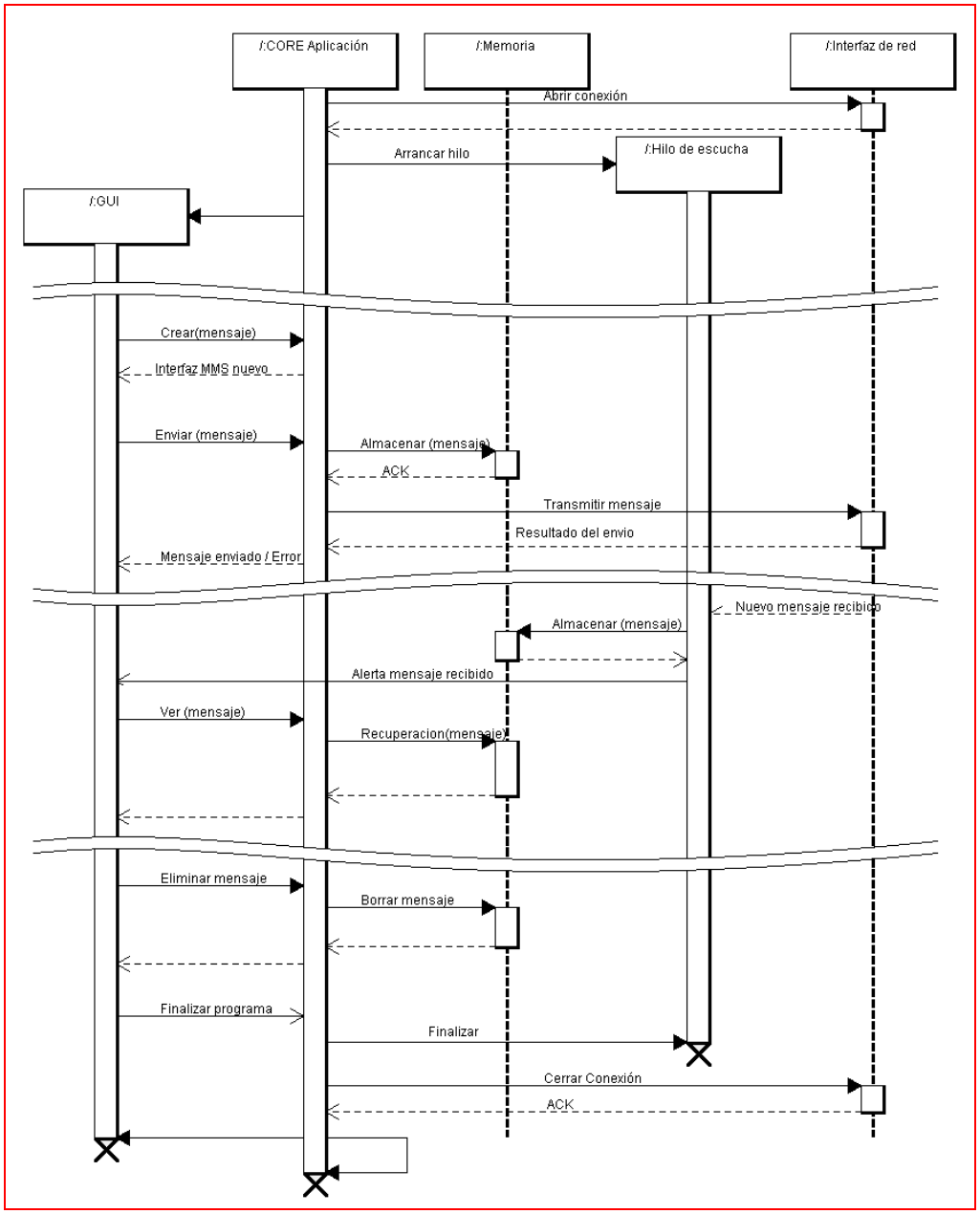


Ilustración 40 - Diagrama de secuencia N800MMS

Estas interacciones son, entre otras:

1. **Arranque**: Al inicio del programa el hilo de ejecución principal debe iniciar el secundario que será el hilo de escucha de mensajes recibidos. También deben cargarse las variables globales y los índices de mensajes en bandeja.

2. **Creación de nuevo mensaje**: El mensaje que el usuario cree en la interfaz gráfica se almacenará en el disco del terminal y se enviará a través de la interfaz wireless y reinforma al usuario del estado del envío del mensaje. En mensaje enviado se indexa y se almacena en la bandeja de enviados.
3. **Recepción de un mensaje**: Cuando el hilo que corre en segundo plano recibe un mensaje, se comunica con el hilo primario para que en la interfaz gráfica aparezca una notificación visual que alerte al usuario de que existe un nuevo mensaje en la bandeja de entrada. El mensaje recibido se almacena e indexa y se guarda, precisamente, en la bandeja de entrada.
4. **Eliminación de un mensaje**: Si el usuario decide eliminar un mensaje de cualquiera de las bandejas, el mensaje es eliminado del disco del terminal y se re-indexan los mensajes existentes.
5. **Finalización de la aplicación**: Antes de cerrar la aplicación el hilo principal finalizará al secundario y se guardará en disco la información relevante de la sesión para dar continuidad a las acciones del usuario cuando vuelva a entrar en la aplicación. A pesar de hacer esta descarga al finalizar la aplicación, durante la ejecución del programa se hacen backup de seguridad de la información relevante tras cada acción del usuario.

4.3.4. COMPONENTES

4.3.4.1. Menu Principal

La interfaz principal (MenuPrincipal.java) presenta por pantalla seis funcionalidades:

1.- Bandeja de entrada: al presionar el botón lanzamos BandejaEntrada.java

2.- Bandeja de salida: al presionar el botón lanzamos BandejaSalida.java

3.- Nuevo MMS: al presionar el botón lanzamos NuevoMMS.java

4.- Logs: al presionar el botón lanzamos Logs.java

5.- Borradores: al presionar el botón lanzamos Borradores.java

6.- Configuración: al presionar el botón lanzamos pantallaConfiguracion.java

Cada uno de los botones abre una nueva pantalla

Antes de lanzar la interfaz gráfica, esta clase se encarga de lanzar el hilo de ejecución paralelo que se mantendrá a la escucha de nuevos mensajes recibidos. También carga las variables globales, los logs y finalmente, lanza la interfaz gráfica.

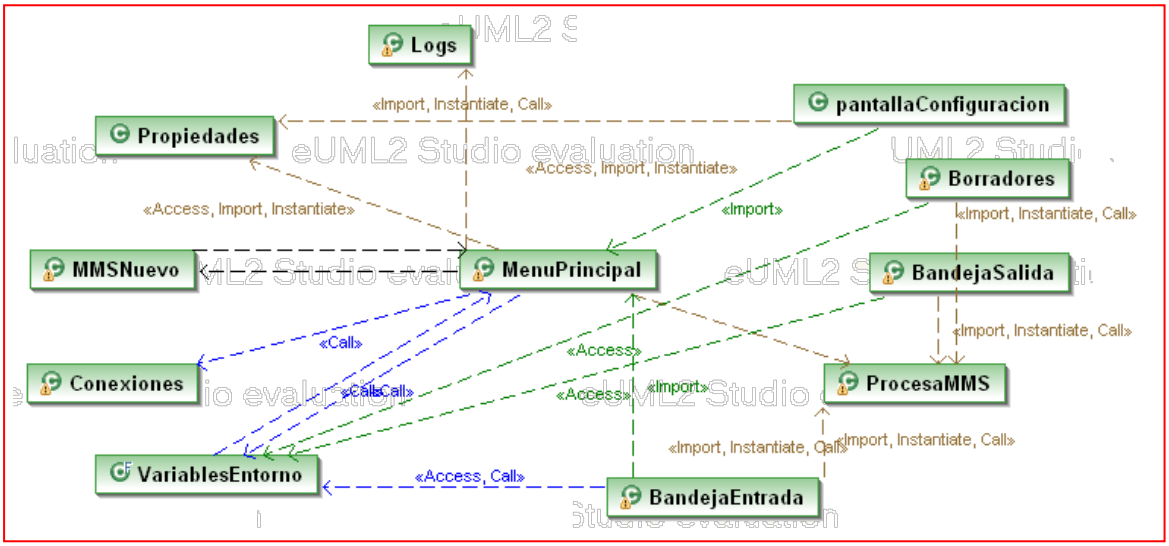


Ilustración 41 - Diagrama de dependencias del Menu Principal

4.3.4.2. Pantallas de Bandeja

Las pantallas de las bandejas se inician desde el menú principal, siguiendo la filosofía de la interfaz de inicio de Maemo. Las pantallas de las bandejas las manejan los fuentes: BandejaEntrada.java, BandejaSalida.java y Borradores.java.

Las bandejas tienen el mismo aspecto, pero enlazan con funcionalidades distintas. Están formadas por varios contenedores:

1. **Contenedor de mensajes**: almacena los mensajes en la bandeja. Este contenedor dispone de una barra de scroll y que puede almacenar un máximo de 50 mensajes (parametrizable).
    a. **Subcontenedor de mensaje**: contiene un mensaje de los almacenados que no son más que una representación gráfica que enlaza con la pantalla de mensaje existente. La información que se muestra en este contenedor es si el mensaje ha sido leído o no: se indicará con el sobre de color amarillo si no ha sido leído y gris al contrario, y el remitente o el destinatario del mensaje, según la bandeja.

2. **Contenedor de previsualización**: Muestra el comienzo del texto del mensaje que se seleccione en el contenedor de mensajes.

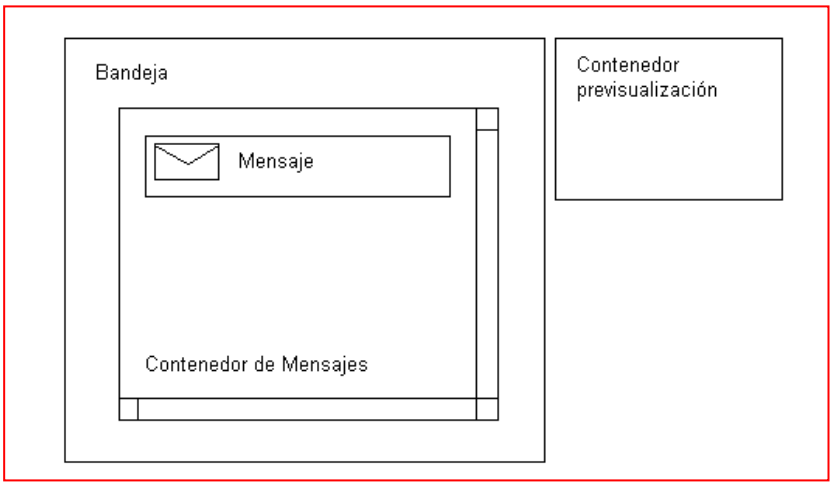


Ilustración 42 - Paneles de contenedores Bandejas

4.3.4.3. Nuevo mensaje

La pantalla de nuevo mensaje consta de dos contenedores principales: botonera y contenedor de mensaje. En el contenedor botonera se ofrecen al usuario las funcionalidades de enviar, de guardar, ir a los logs, ver videos y volver a la pantalla principal.

En el contenedor de mensaje existen los subcontenedores de Para, de Asunto, de texto, Contado de caracteres del subcontenedor de texto, un contenedor botonera para la gestión de adjuntos, un contenedor de previsualización de imágenes y un contenedor para alojar los archivos adjuntos.

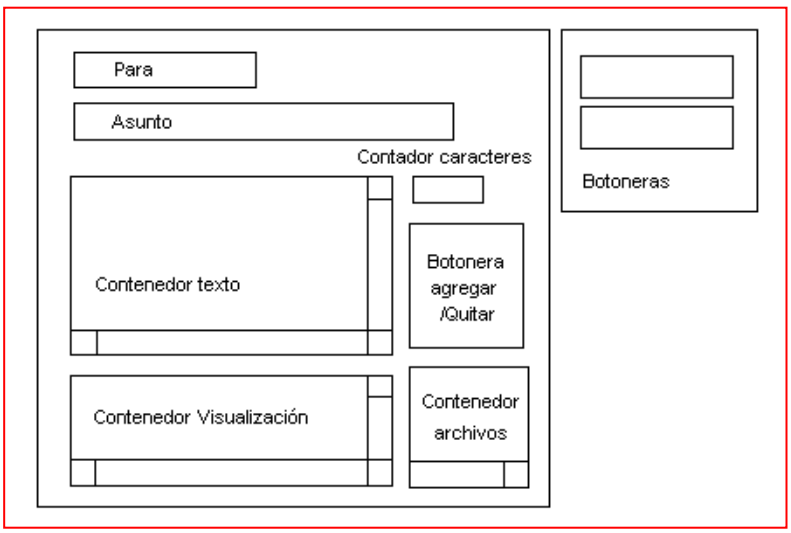


Ilustración 43 - Paneles Mensaje Nuevo

Cuando se selecciona en el contenedor de archivos un archivo de imagen, la imagen se vuelca al contenedro de previsualización. La imagen puede agrandarse haciendo doble click en el contenedor. Si el archivo seleccionado es un archivo de audio se reproducirá mediante las librería de MP3 para java. Si el archivo seleccionado es de video se podrá visualizar mediante el botón de Ver Videos que hará una llamada al reproductor de video nativo del sistema.

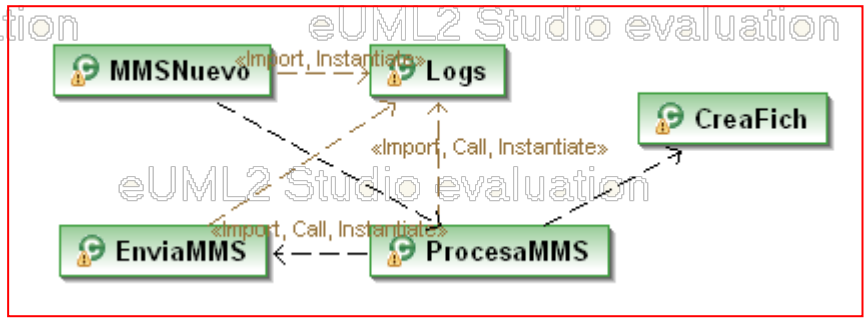


Ilustración 44 - Diagrama de dependencias Nuevo mensaje

4.3.4.4. Envío y gestión de mensajes

El envío de mensajes se delega en la clase ProcesaMMS.java que gestiona los índices de los mensajes guardados en disco, gestiona el envío, almacena y borra MMS.

El envío de MMS lo hace la clase EnviaMMS.java que gestiona con el MMSClientEnv.java el envió a través de la interfaz wireless. La clase MMSClientEnv.java es la que codifica el mensaje mediante las MMSLibraries de Nokia conforme al estándar WAP-209.

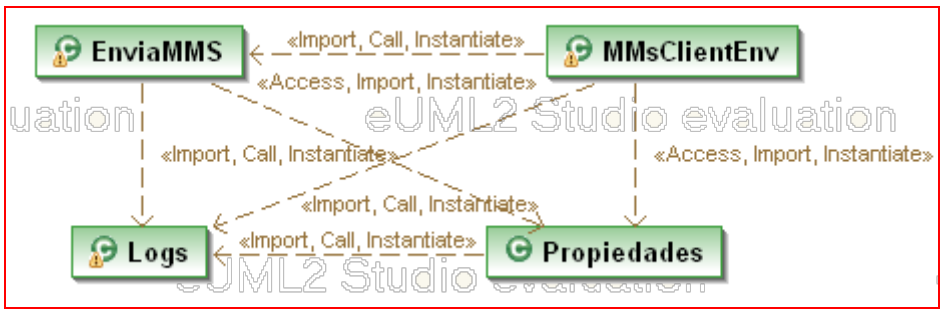


Ilustración 45 - Diagrama de dependencias Envío de mensajes

En la recepción de mensajes el hilo MMSClientRec.java, de ejecución paralela al principal, pasa el mensaje recibido a la clase ProcesaMMS.java para que decodifique con las MMSLibraries y almacene en disco el mensaje recibido.

Asimismo se notifica al hilo principal que se ha recibido un mensaje para que pinte un sobre amarillo de alerta en la zona habilitada al uso de la pantalla principal y emita una alerta sonora para atraer la atención del usuario.

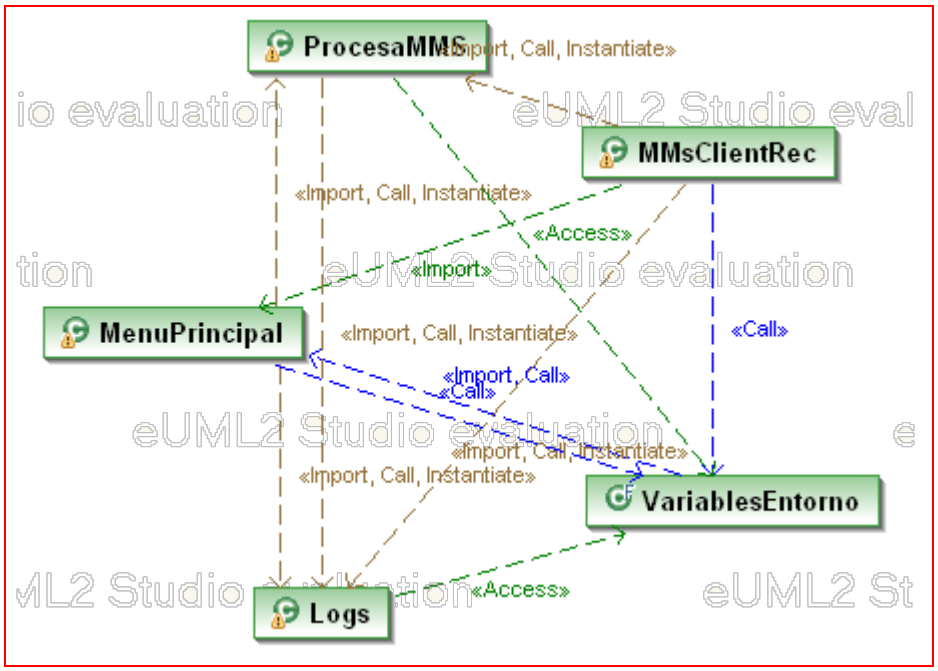


Ilustración 46 - Diagrama de dependencias de mensaje recibido

4.3.4.4.1. Codificación y decodificación

La codificación y decodificación de mensajes se realiza de acuerdo con el estándar WAP-209 mediante el uso de MMS Java Libraries de Nokia.

```
…
      mm.setVersion(IMMConstants.MMS_VERSION_10);
      mm.setMessageType(IMMConstants.MESSAGE_TYPE_M_SEND_REQ);
      mm.setTransactionId(id);
      mm.setDate(new Date(System.currentTimeMillis()));
      if (para.indexOf('@') != -1) {
        mm.addToAddress(para);
      } else {
        mm.addToAddress("+34" + para + "/TYPE=PLMN");
      }
      mm.setFrom("+34"+numTerminal+"/TYPE=PLMN");
      mm.setDeliveryReport(false);
      mm.setReadReply(false);
      mm.setSenderVisibility(IMMConstants.SENDER_VISIBILITY_SHOW);
      mm.setSubject(asunto);
      mm.setMessageClass(IMMConstants.MESSAGE_CLASS_PERSONAL);
      mm.setPriority(IMMConstants.PRIORITY_LOW);
      mm.setContentType(IMMConstants.CT_APPLICATION_MULTIPART_MIXED);
…
      MMContent part1 = new MMContent();
      byte[] buf1 = textoMMS.getBytes();
      part1.setContent(buf1, 0, buf1.length);
      part1.setContentId("<" + contador + ">");
      part1.setType(IMMConstants.CT_TEXT_PLAIN);
      mm.addContent(part1);
      …
      MMContent part1 = new MMContent();
      byte[] buf1 = readFile((String) archivos.get(i));
      part1.setContent(buf1, 0, buf1.length);
      part1.setContentId("<" + contador + ">");
```

```
      part1.setType(recuperarFormato((String) archivos.get(i)));
      mm.addContent(part1);
      …

      recuperarFormato:

      if (path.endsWith(".gif")) {
        return IMMConstants.CT_IMAGE_GIF;
      } else if (path.endsWith(".jpg") || path.endsWith(".jpeg")) {
        return IMMConstants.CT_IMAGE_JPEG;
      } else if (path.endsWith(".mp3")){
        return "audio/mpeg";
      } else if (path.endsWith(".wav")){
        return "audio/x-wav";
      } else if (path.endsWith(".avi")){
        return "video/x-msvideo";
      }

      …

private boolean parseRequest() {
       byte[] match = { 13, 10, 13, 10 };
       int pos, ch, contentLength;
       int index = 0;
       String line = "";
       Logs logs = new Logs();
       try {
              while ((ch = in.read()) >= 0) {
                     if (ch == match[index] && index <= 3)
                            index++;
                     else
                            index = 0;
                            if (index == 4) {
                            // Gets the request content.
                     contentLength = Integer.parseInt((String) headers
                                                   .get("content-length"));
                     logs.writeLog("taman:" + contentLength);
                     byte[] content = new byte[contentLength];
                     for (int j = 0; j < contentLength; j++)
                            content[j] = (byte) in.read();
                            MMDecoder mmDecoder = new MMDecoder();
                            mmDecoder.setMessage(content);
                            mmDecoder.decodeMessage();
                            MMMessage mm = mmDecoder.getMessage();
                            logs.writeLog("Decodificado mensaje");
                            printMessage(mm);
                            storeContents(mm);
                            ProcesaMMS pp = new ProcesaMMS();
                            break;
                     }
                     // Gets the request headers
                     line += (char) ch;
                     if ((line.length() > 2) && (ch == '\n')) {
                            pos = line.indexOf(':');
                            if (pos != -1)
                                   setHeader(line.substring(0, pos).toLowerCase(),
                               line .substring(pos + 2, line.length() - 2));
                            else
                               setRequestType(line.substring(0, line.length() - 2));
                               line = "";
                            }
              }
              return true;
       } catch (Exception e) {
              logs.writeLog(e.getMessage());
              return false;}}
```

Tabla 18 - Codificación y decodificación de mensajes

4.3.4.4.2. Envío de MMS

El envío de mensajes se realizar a través de MMSLibrary de Nokia

```
try {
        MMEncoder encoder = new MMEncoder();
        encoder.setMessage(mms);
        encoder.encodeMessage();
        byte[] out = encoder.getMessage();
        MMSender sender = new MMSender();
        Propiedades p = new Propiedades();
        sender.setMMSCURL(p.URLMMSC.trim());
        MMResponse response = sender.send(out);
        logs.writeLog("Mensaje enviado a: " + sender.getMMSCURL());
        logs.writeLog("Codigo de respuesta: "
                        + response.getResponseCode() + " "
                        + response.getResponseMessage());
        Enumeration keys = response.getHeadersList();
        while (keys.hasMoreElements()) {
                String key = (String) keys.nextElement();
                String value = (String) response.getHeaderValue(key);
                logs.writeLog(key + ": " + value);
        }
} catch (Exception e) {
        e.printStackTrace();
}
```

Tabla 19 - Envio de mensajes con librerías Nokia

4.3.4.4.3. Recepción MMS

La recepción de mensajes se realizar a través de MMSLibrary de Nokia.

```
…
while (true) {
        Socket incoming = s.accept();
        VariablesEntorno.getmrx();
        in = new DataInputStream(incoming.getInputStream());
        out = new DataOutputStream(incoming.getOutputStream());
        client = incoming.getInetAddress().getHostAddress() + ":"
                + HOST_PORT;
        logs.writeLog("Nuevo mensaje recibido de: " + client);
        String response;
        // Decodes the message and returns a response to the sender.
        boolean success = parseRequest();
        logs.writeLog("Parseando mensaje");
        if (success)
        response = "HTTP/1.1 "+ (mode == SYNC_MODE ? "204 No Content"
                                        : "202 Accepted") + "\r\n";
        else
        response = "HTTP/1.1 400 Message Validation Failed\r\n";

        out.writeBytes(response);
        out.flush();
        out.close();
        in.close();
        incoming.close();
        if (success && mode == ASYNC_MODE) {
                Thread.sleep(RESPONSE_DELAY);
                sendAsyncResponse();
        }

}

…
```

Tabla 20 - Recepción de mensajes con librerías Nokia

4.3.4.4.4. Gestión MMS en el sistema

En el procesado de mensajes los MMS se almacenan en disco y se gestionan mediante un sistema de índices asociados a bandejas. Los índices se actualizan cada vez que se hace cualquier acción sobre los mensajes: guardar, enviar, recibir o eliminar.

El contenido del mensaje multimedia se almacenará en el sistema como diferentes archivos con un enlace común al identificador de mensaje, así si el mensaje mms001 de la bandeja tiene texto y diferentes adjuntos como imágenes y video, en el archivo de descripción del mensaje mms001 se guardará la referencia de la ubicación (directorio donde han sido almacenados) de los archivos adjuntos en el sistema.

Así mismo, al almacenar o borrar un mensaje, además de hacerlo en el soporte físico del dispositivo, se actualizará el fichero de indexación de mensajes por bandeja que contiene la referencia a los archivos de descripción de los mensajes y la ubicación en lo que a bandejas se refiere.

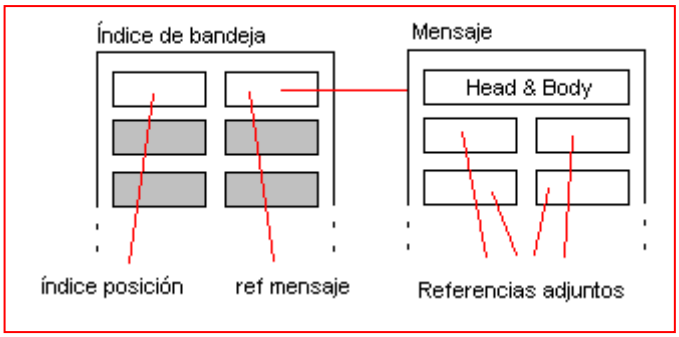


Ilustración 47 - Indexación mensajes

4.3.4.5. Logs

Todas las pantallas cuando realizan acciones solicitadas por el usuario arrojan logs en el fichero indicado al uso que se guarda en disco para que los desarrolladores puedan ver los posibles errores o trazas. Se presta especial interés a los logs de las clases de envío, codificación, decodificación y recepción.

En la pantalla de logs se muestra el texto de este archivo.

4.3.4.6. Configuración

En la pantalla de configuración se puede cambiar el puerto de escucha del hilo de recepción y la dirección de envío del hilo de transmisión.

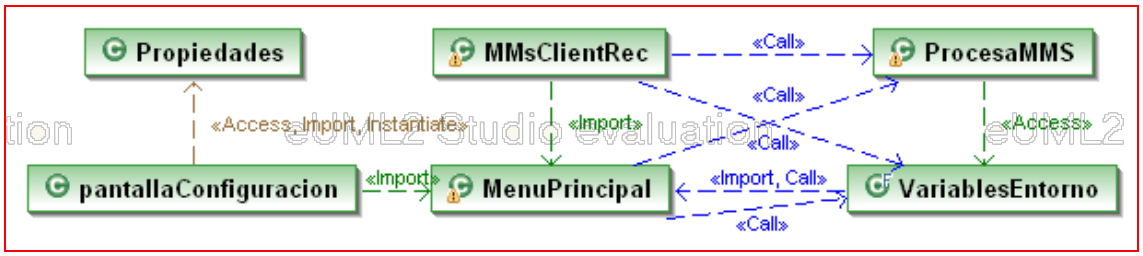


Ilustración 48 - Diagrama de dependencias pantalla de Configuración

El usuario puede cambiar dos puntos fundamentales en la configuración de la aplicación tal y como hemos comentado anteriormente. Si el usuario cambia el puerto de escucha, el programa reinicia el socket de recepción. Si el usuario cambia la dirección de envío este valor se cambia en las variables globales del sistema y se guarda en la configuración por defecto de la aplicación el nuevo valor.

4.4. Ajustes de entornos

En este proyecto se han descrito los pasos para poder montar el entorno de desarrollo en sistemas Linux o Maemo sobre Linux. Todas las características visuales de la aplicación han sido adaptadas para optimizarlas en dispositivos N800, tal y como comentábamos en el apartado del estado del arte en la sección de la usabilidad para dispositivos móviles, no obstante, si quisiésemos trabajar/desarrollar en entornos windows o WM6, puesto que nuestra aplicación lo permite, simplemente tendríamos que ajustar el fichero de propiedades de entornos, entornoWIN.txt y entornolinux.txt que dan la ruta de la aplicación en el sistema: por ejemplo /home/user/n800mms o bien C:\n800mms, y descargarnos el paquete swt apropiado para el SO [30].

4.4.1. WINDOWS MOBILE

Al igual que en Maemo, Jalimo está disponible para la plataforma WM6. Para instalar Jalimo en windows Mobile 6 seguiremos las siguientes instrucciones:

Instalamos phoneME Advanced Dual Stack - Foundation b114 (with MIDP support)(cab) desde:http://www.cs.kuleuven.be/~davy/phoneme/index.php?q=node/10 en nuestro dispositivo móvil y creamos un lnk de nuestra aplicación para probarlo con la secuencia de arranque:

[30] Disponible en http://www.eclipse.org/downloads/download.php?file=/eclipse/downloads/drops/R-3.5.1-200909170800/swt-3.5.1-win32-win32-x86.zip

```
254#"\Program Files\pMEA FP\bin\cvm.exe" -cp "\jalimo\swt.jar;\jalimo\jalimo.jar" "-Djava.library.path=\jalimo" org.jalimo.examples.swt.SwtSample
```

Para instalar la aplicación N800MMS

1.- Copiamos el contenido de la carpeta WM6N800MMS al root de nuestro dispositivo.

2.- Creamos el nuevo lnk n800mmsWM6:

```
254#"\Program Files\pMEA FP\bin\cvm.exe" "-Xopt:stdioPrefix=/storage card,useConsole=false" -cp "\n800mms\N800MMS.jar;\n800mms\MMSLibrary.jar;\n800mms\swt-gtk.jar;\n800mms\jl1.0.jar:" "-Djava.library.path=\n800mms" MenuPrincipal
```

5. PRUEBAS

Tras finalizar cada línea de desarrollo se somete el código a pruebas unitarias, de tal forma que se pueda garantizar el correcto funcionamiento de cada módulo específico a la hora de la integración. Cuando los módulos específicos se han desarrollado y testado, se unen para conformar la aplicación y se procede a pasar el *test* de pruebas integradas. Del análisis e implementación de los *test* de pruebas unitarias y pruebas integradas se genera el *test* de pruebas de regresión que será necesario pasar tras la solución de cualquier *bug* detectado en la fase de pruebas de usuario (UAT).

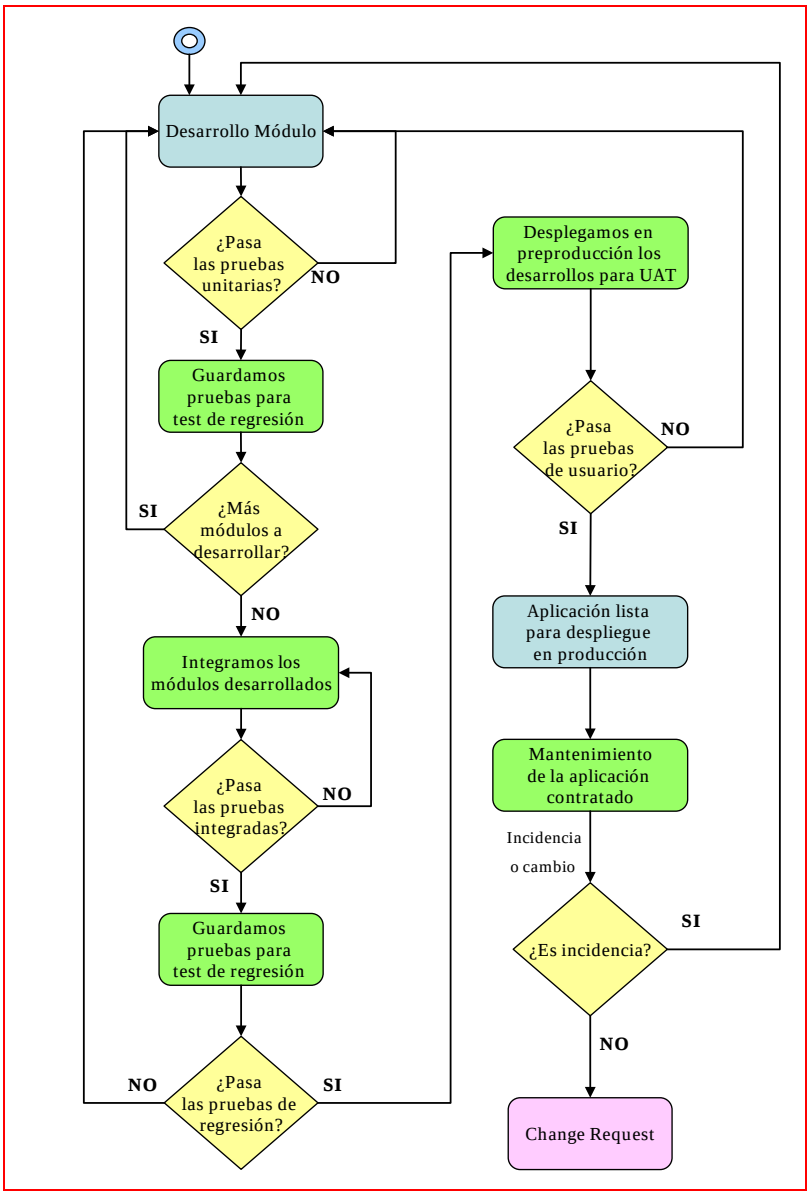


Ilustración 49 - Flujo de pruebas de la aplicación

5.1. Escenarios de Pruebas

Definiremos tres escenarios de pruebas según el punto en el que nos encontremos dentro del flujo de pruebas de la aplicación: unitarias, integradas y UAT. En este apartado describiremos las pruebas unitarias e integradas, puesto que el plan de pruebas de la fase de UAT depende exclusivamente del usuario o cliente, siendo nuestra implicación en estas últimas simplemente de soporte.

5.1.1. PRUEBAS UNITARIAS

El primero de los escenarios de pruebas será el propio Eclipse, la herramienta de desarrollo. Mediante JUnit, integrado en Eclipse, diseñaremos los *test* de pruebas de los bloques de código de funcionalidad paralela mientras vamos desarrollando. Todas las pruebas que vayamos recopilando, fruto del desarrollo, servirán posteriormente como pruebas de regresión.

5.1.1.1. JUnit

Existe una metodología de desarrollo de programas denominada *Extreme Programming* [2] que está principalmente enfocada en la prácticas y técnicas de pruebas unitarias. Se han creado frameworks específicos en diferentes lenguajes de programación para ayudar precisamente a la realización de esta tarea. Al conjunto de estos frameworks se les ha denominado xUnit.

JUnit es el xUnit para Java, es un Framework Open Source para la automatización de las pruebas (tanto unitarias, como de integración) en los proyectos Software. El framework provee al usuario de herramientas, clases y métodos que le facilitan la tarea de realizar pruebas en su sistema y así asegurar su consistencia y funcionalidad.

En xUnit nos encontramos con dos conceptos clave: los test cases o casos de prueba, que son módulos que ejecutan los métodos de las clases que son objeto de las pruebas, y los test suites o las suites de prueba, que agrupan conjuntos de casos de prueba sobre clases que están funcionalmente relacionadas. Las suites a su vez pueden estructurarse en árboles, siendo los test cases las hojas de dichos árboles.

Mediante estas estructuras se consigue sistematizar los programas de pruebas apoyándonos en este framework que permite que la implementación de las pruebas esté codificada de tal manera que puedan ejecutarse una y otra vez, lo cual es muy útil a la hora de realizar pruebas de regresión. Estas estructuras son también muy útiles a la hora

de realizar pruebas por secciones, ya que, además de poder realizar pruebas a todo el sistema, permiten ejecutar pruebas relativas sólo a una parte del sistema.

5.1.2. PRUEBAS INTEGRADAS

En los test de integración, comprobaremos todas las funcionalidades descritas en los casos de uso de la herramienta basándonos en los test desarrollados mediante JUnit y en las pruebas directas del desarrollador.

Por otro lado, basándonos en programas de grabación de macros, como *Workspace Macro, Easy Macro Recorder* o similares, someteremos a la aplicación a pruebas de estrés. Para ver la reacción del programa y su rendimiento cuando debe enviar, recibir y procesar numerosos mensajes y acciones de usuario.

Finalmente, en esta fase de pruebas integradas vamos a probar la funcionalidad de la aplicación a nivel de protocolos de envío por la red para confirmar que los mensajes emitidos por la aplicación son estructural y formalmente correctos. Para este último punto utilizaremos un sniffer, *Ethereal*.

PRUEBAS
UNITARIAS
Módulo Envío
Modulo Recepción
Módulo almacenamiento
Módulo gestor de índices
Módulo de logs
Módulo de configuración
Módulo gestor de interrupciones
INTEGRADAS
Test casos de uso definidos
Rendimiento Interfaz gráfica
Rendimiento envío y recepción de mensajes
Formato en mensajes adaptado al estándar

Tabla 21 - Enumeración Pruebas Unitarias e Integradas realizadas

5.1.2.1. Capturas de tráfico con Ethereal

Ethereal es un sniffer bastante conocido y que podemos obtener de forma gratuita de la web oficial [31]. Se conoce como sniffing a la captura de la información que pasa por la red. Esta aplicación es capaz de capturar todos los paquetes de información que

[31] Sitio web oficial ***Ethereal***: http://www.ethereal.com

se difunden a través de la red, escuchando por el interfaz que le indiquemos, para posteriormente interpretarlos.

Para capturar el tráfico de la aplicación vamos a utilizar un par de portátiles donde ejecutaremos el programa.

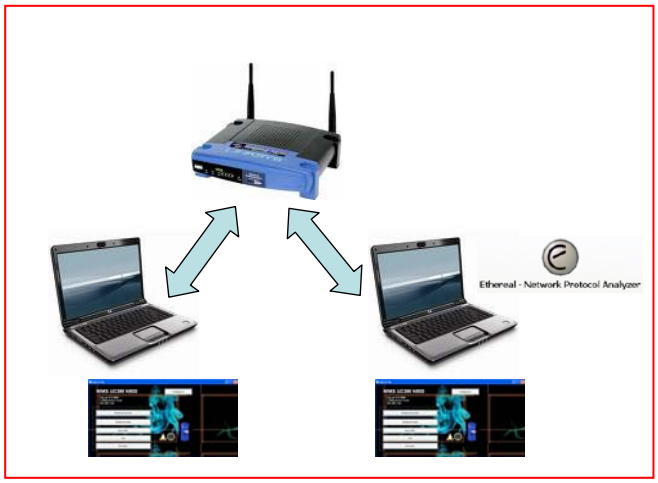


Ilustración 50 - Escenario para captura de tráfico

Configuraremos los programas para que la dirección de envío de los mensajes se efectúe entre ambos PC's:

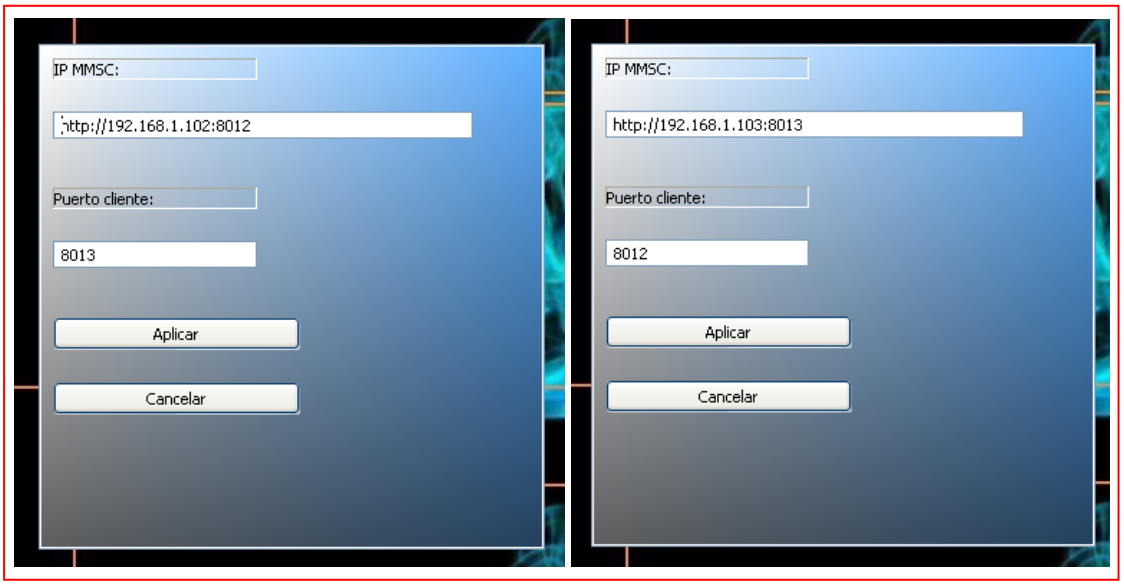


Ilustración 51 - Configuración programas en PC's para pruebas de captura de tráfico

Una vez tenemos arrancados los dos programas, en uno de los portátiles arrancamos el programa Ethereal y lo ponemos en modo promiscuo a través de la interfaz wireless:

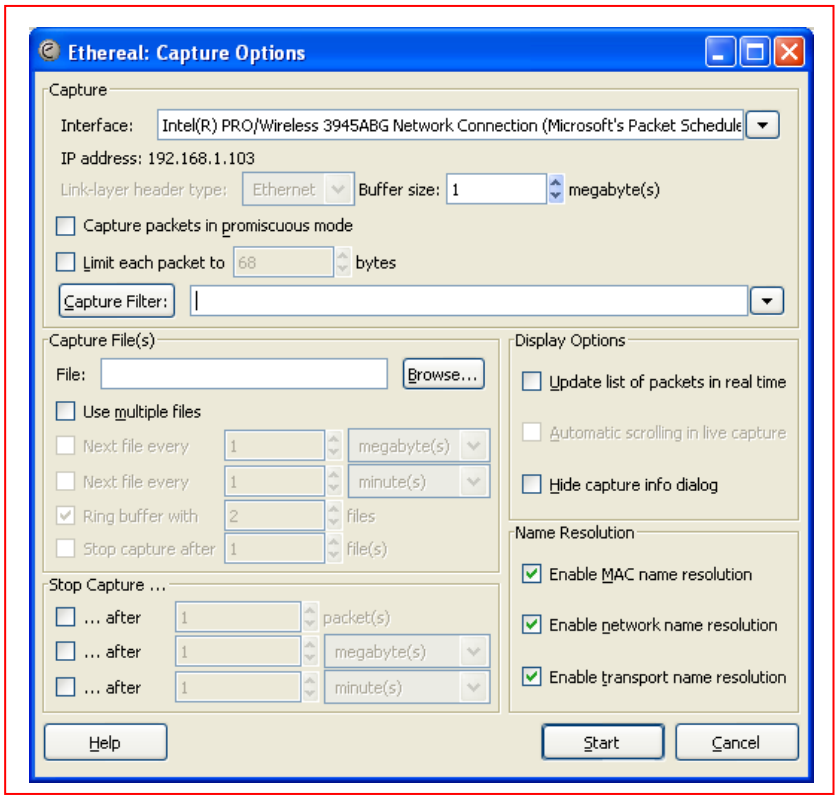


Ilustración 52 - Configuración de escucha del programa Ethereal

A continuación procedemos a enviar un mensaje desde la aplicación:

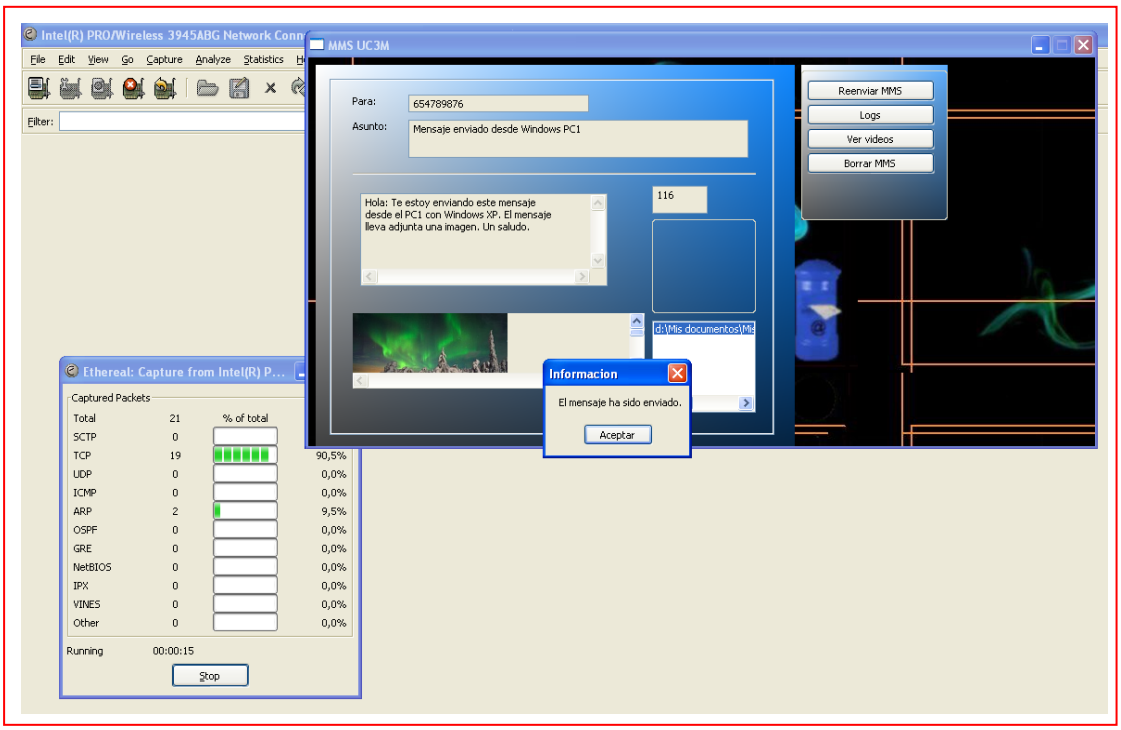


Ilustración 53 - Captura de paquetes de un mensaje enviado

Y a recibir un mensaje enviado desde otro terminal:

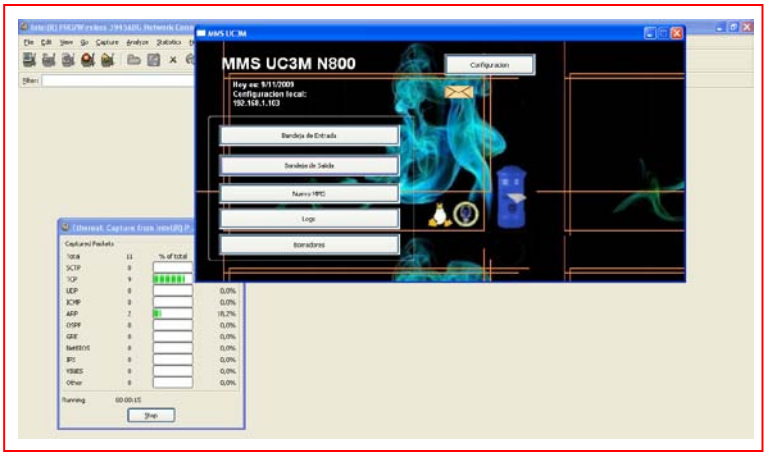


Ilustración 54 - Captura de paquetes de un mensaje recibido

Una vez enviado y recibido los mensajes, paramos la escucha de paquetes y procedemos a analizar el tráfico capturado.

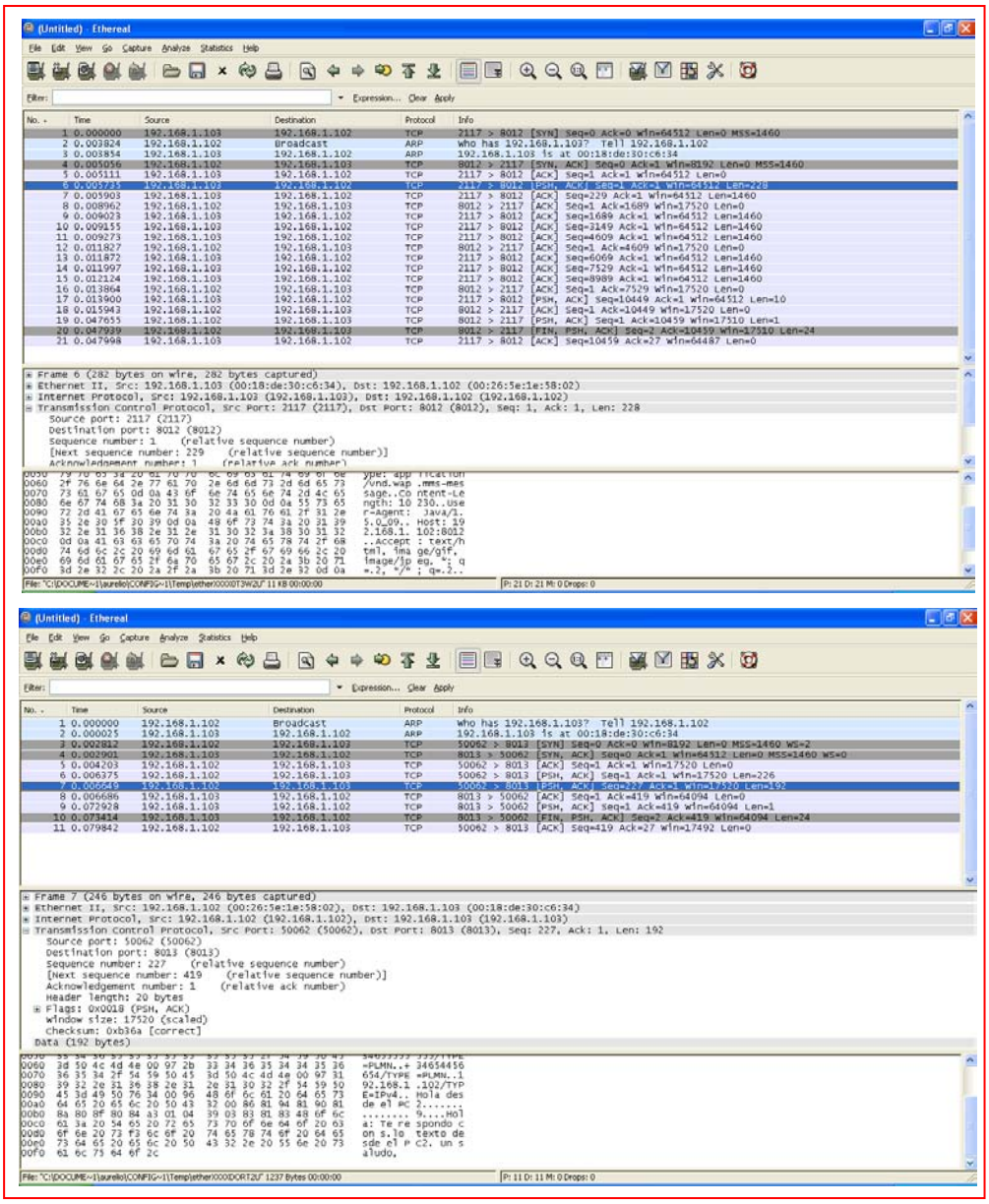


Ilustración 55 - Paquetes capturados con Ethereal de un MMS enviado, arriba, y recibido, abajo.

Como podemos observar, en las capturas, las primeras tramas son de tipo ARP (Address Resolution Protocol). El protocolo de resolución de direcciones es el responsable de convertir la dirección IP a la dirección de red física. Podemos ver a continuación las tramas ARP Request y Replay:

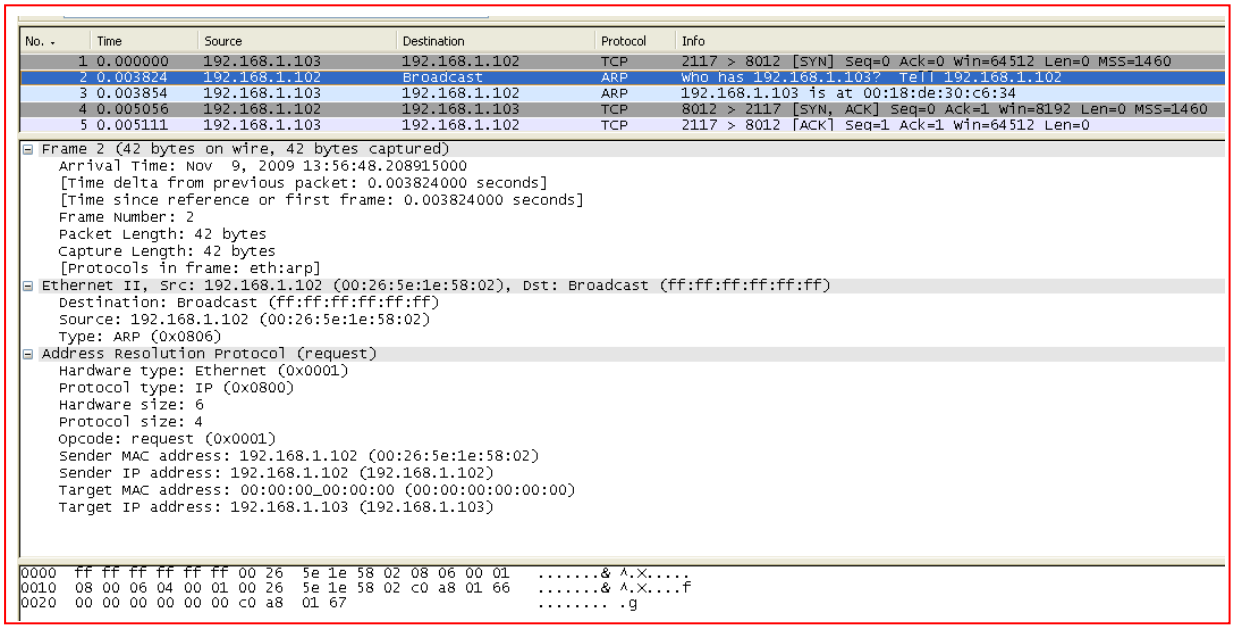


Ilustración 56 - Trama ARP Request

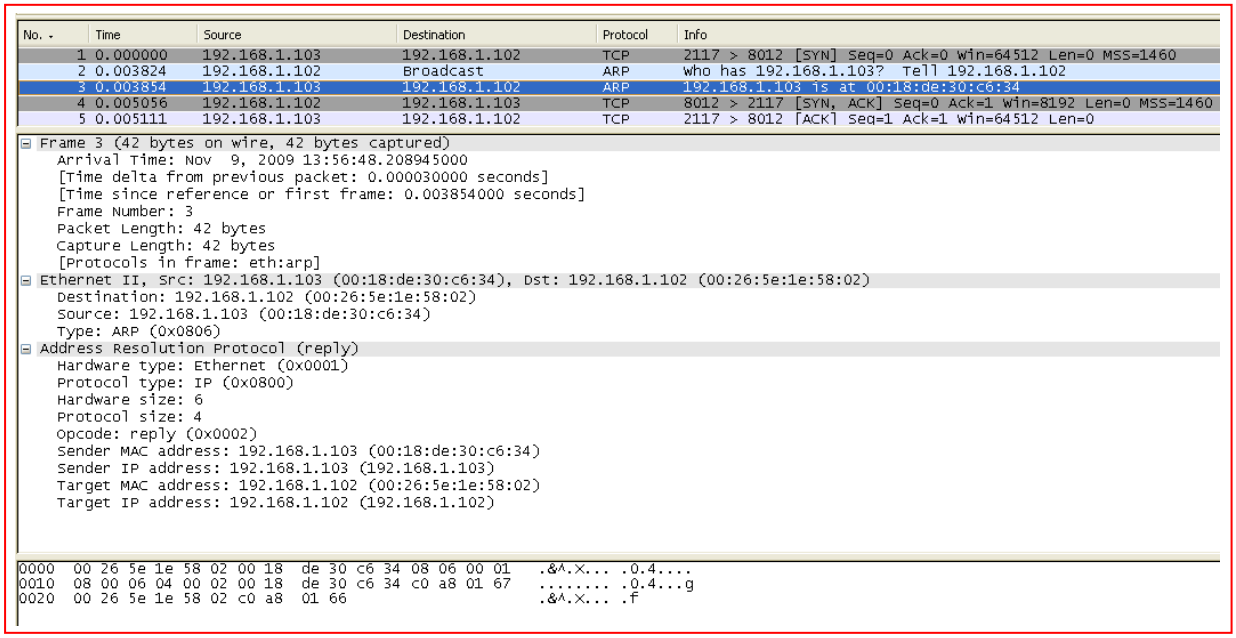


Ilustración 57 - Trama ARP Replay

Si nos fijamos en las tres primeras tramas TCP de la comunicación observaremos el establecimiento de la comunicación, conocido como *three wayhandshaking*.

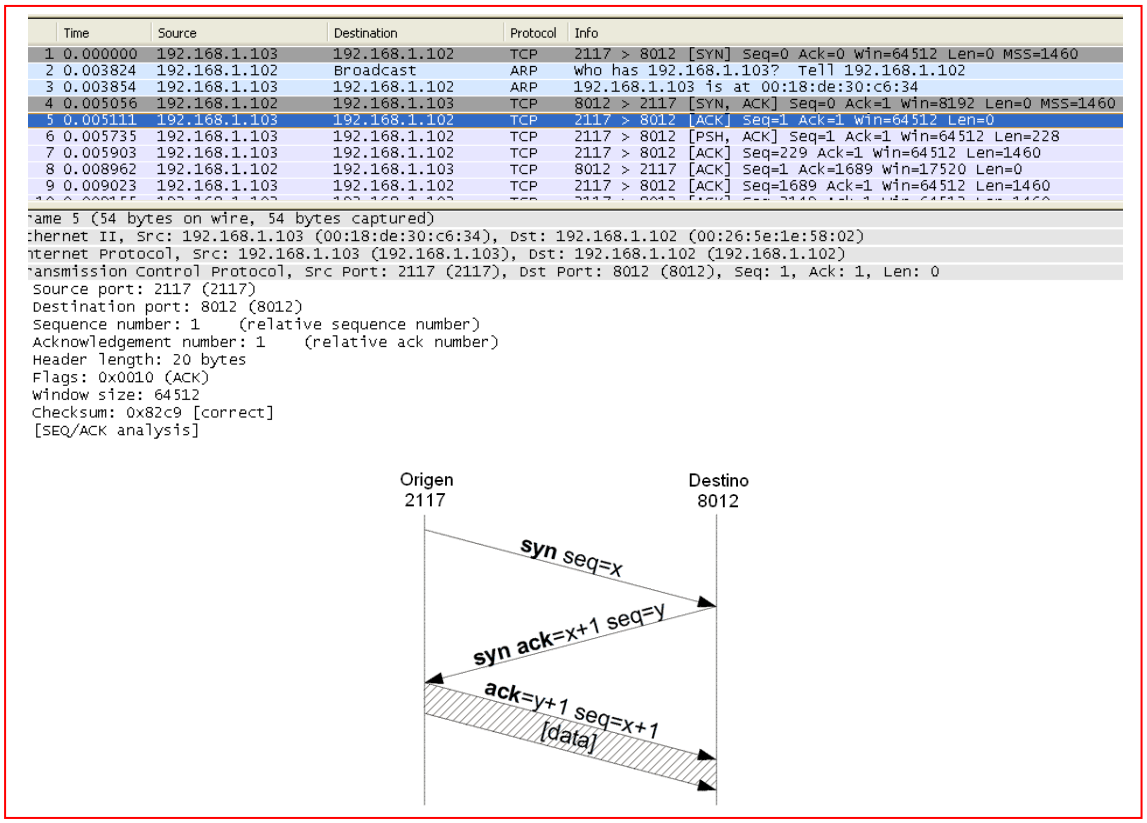


Ilustración 58 - Establecimiento de la comunicación TCP

A continuación se empiezan a transmitir los datos según la trama MMS:

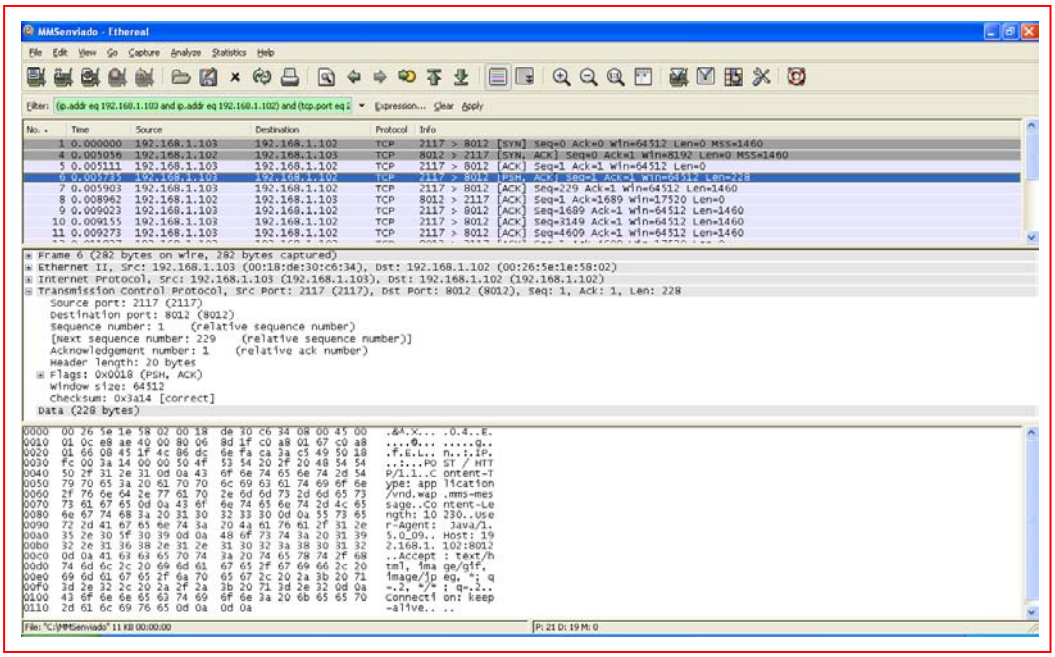


Ilustración 59 - Parte inicial mensaje MMS

La estructura de la PDU (Protocol Data Unit) de MMS está especificada por el WAP Forum en la especificación WAP-209-MMSEncapsulation. El contenido de dicha PDU (que podemos ver en la imagen 82 en formato ASCII) se distribuye como se especifica en la siguiente figura.

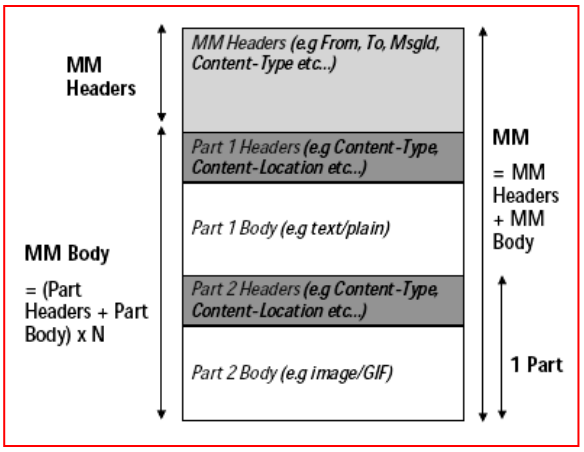


Ilustración 60 - Estructura PDU de MMS [18]

En la parte de los MM-Headers encontramos información específica de la PDU sobre cómo transferir la información del medio de origen al destino. Por ejemplo, podemos ver el tipo del contenido, la longitud, etc… En la parte del cuerpo del mensaje, tenemos de nuevo una parte de cabecera que nos indica información sobre la transacción mms que se está realizando, por ejemplo: ID Transacción: 0000000066, de quién proviene el mensaje: +34655555555/TYPE=PLMN, a quien va dirigido: +34654789876/TYPE=PLMN, el asunto: Mensaje enviado desde Windows PC1, etc (ver apartado 4.4.4.4.1 de codificación del mms).

Y finalmente se transmiten las diferentes partes del mensaje: el texto plano, las imágenes, los videos…

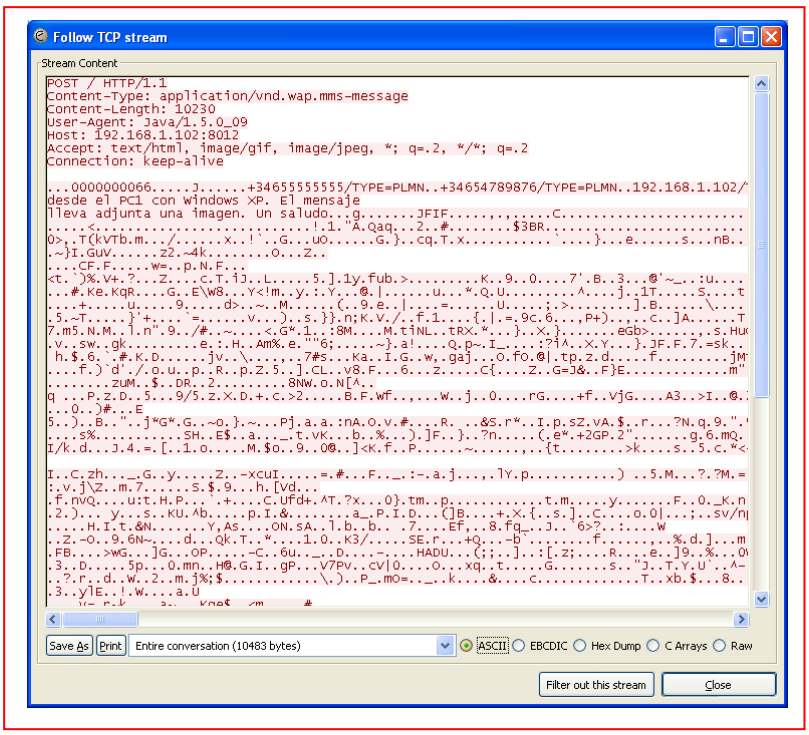


Ilustración 61 - Seguimiento del paquete en formato ASCII.

Al finalizar la transmisión, se recibe un mensaje del tipo: HTTP/1.1 204 No Content. Este mensaje indica que la operación se ha realizado con éxito.

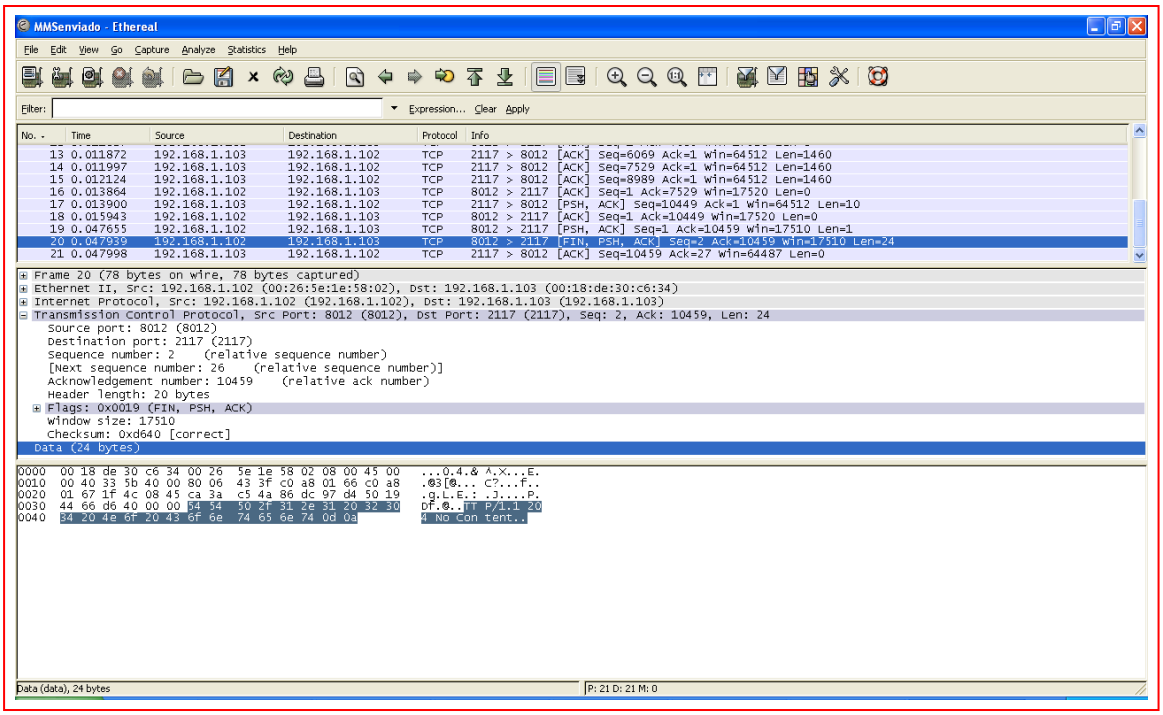


Ilustración 62 - Finalización del mensaje con éxito

6. ENTREGABLES

6.1. MANUAL DE USUARIO

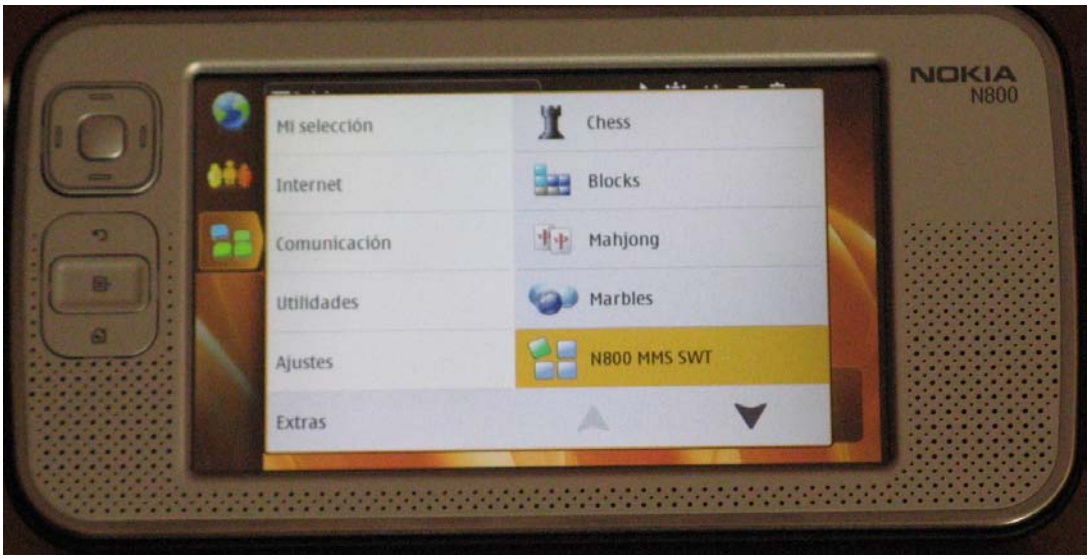
6.1.1. INTRODUCCIÓN

El objeto de este documento es la elaboración del manual de usuario del aplicativo MMS N800 UC3M.

La aplicación MMS N800 está diseñada para que su uso sea intuitivo y sencillo. En este manual explicaremos algunos de los aspectos de la herramienta para facilitar la navegación a través de ella.

6.1.2. MANUAL

Para abrir la herramienta vaya al menú de la izquierda, pinche sobre el icono de herramientas y vaya a Extras allí aparecerá la herramienta MMS N800 UC3M:



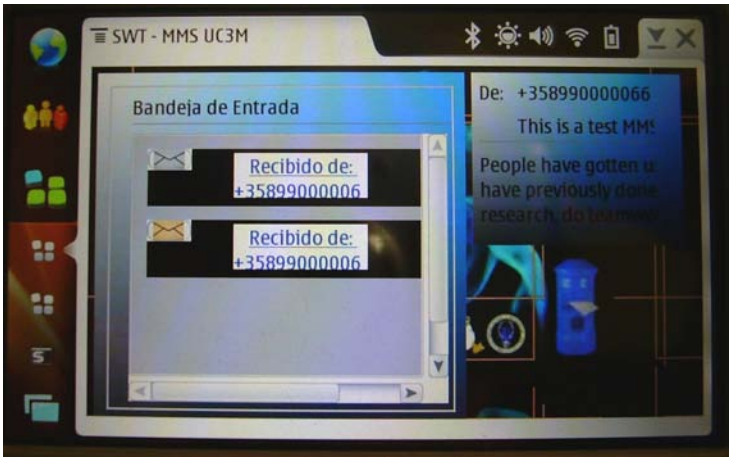
Tardará algunos segundos en cargar, espere hasta que aparezca en la pantalla la aplicación.

Una vez dentro de la herramienta, observaremos las mismas utilidades que en cualquier otra aplicación MMS.


Cuando reciba un mensaje sonará un aviso y aparecerá un sobre amarillo como recordatorio de que ha recibido un MMS. Para ver el mensaje y que desaparezca el sobre, entre en la bandeja de entrada.

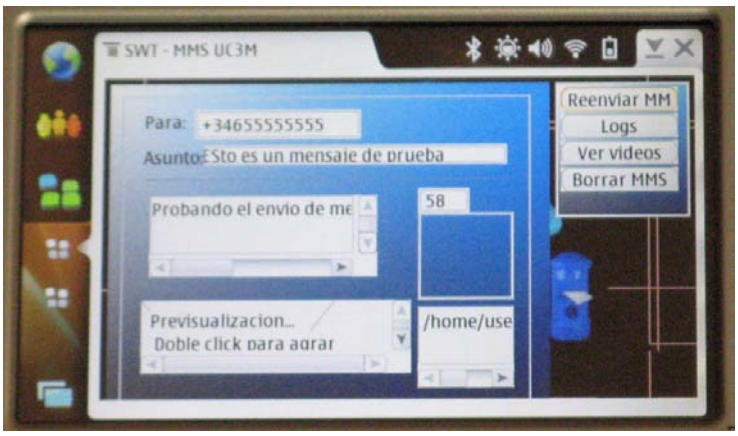
6.1.2.1. Bandeja de entrada

En ella se almacenarán los mensajes que reciba la N800. Los mensajes que ya haya leído aparecerán marcados con un sobre en tono gris, mientras que los que aún no haya leído aparecerán marcados en amarillo.



Si pincha sobre el texto que aparece en cada mensaje podrá ver una pre-visualización del texto a la derecha. Si quiere ver el mensaje completo, pinche en el sobre y se abrirá el mensaje.

Con un mensaje recibido, usted puede reenviar en MMS, borrarlo, ver videos adjuntos si el mms tiene alguno y, en modo desarrollador de la aplicación, ver los logs de la aplicación.



El área de visualización de fotos adjuntas aprovecha la pantalla táctil. Al doble clic sobre esta área, la foto se mostrará a pantalla completa.

La visualización de videos se hace mediante el reproductor de video nativo de la N800. Para visualizar un video adjunto, selecciónelo en el área de adjuntos y presione el botón “Ver videos”.

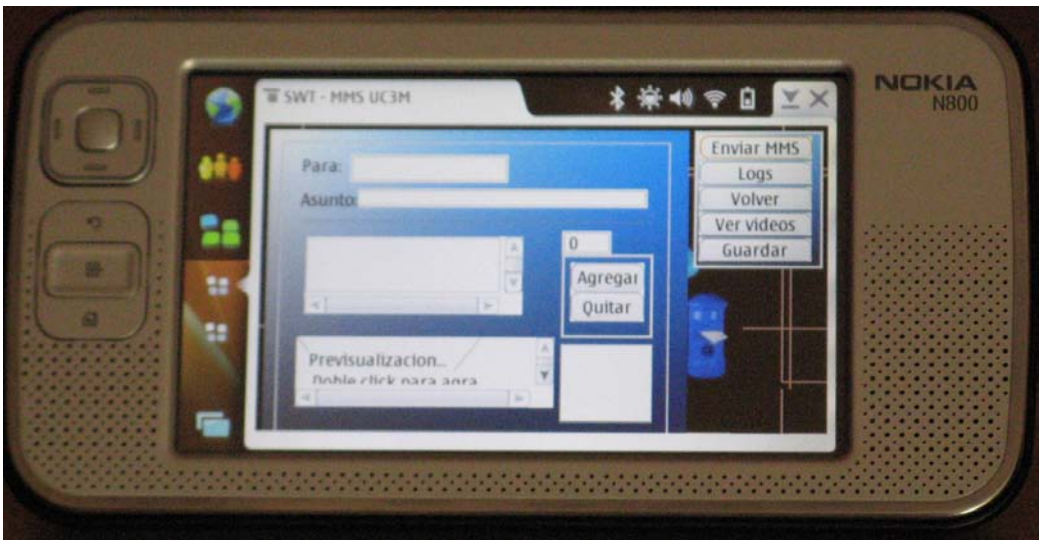
6.1.2.2. Bandeja de salida

Todos los mensajes que enviemos o reenviemos se almacenarán en esta bandeja. El funcionamiento de la bandeja de salida es la misma que la de la de entrada.

6.1.2.3. Nuevo MMS

Al seleccionar esta funcionalidad aparecerá la pantalla de nuevo mensaje.

Los campos a rellenar son los habituales en cualquier servicio de mensajería: Para (número de teléfono o mail), Asunto y Texto a mandar. También puede agregar/quitar imágenes, videos o fotos a su mms. Puede previsualizar sus fotos pinchando sobre la foto que se quiera ver en la lista que aparece debajo de los botones. Aparecerá en la ventana de abajo a la izquierda, si desea verla en mayor tamaño haga doble click sobre la ventana donde se está mostrando la foto.



El campo Para será útil si en el campo MMSC del menú configuración está informado con la dirección de un centro de mensajería que será el encargado de resolver la IP del terminal al que corresponde el número de teléfono. Si no el mms será enviado al terminal que se designe en el MMSC del menú de configuración que puede designar a un terminal en concreto.

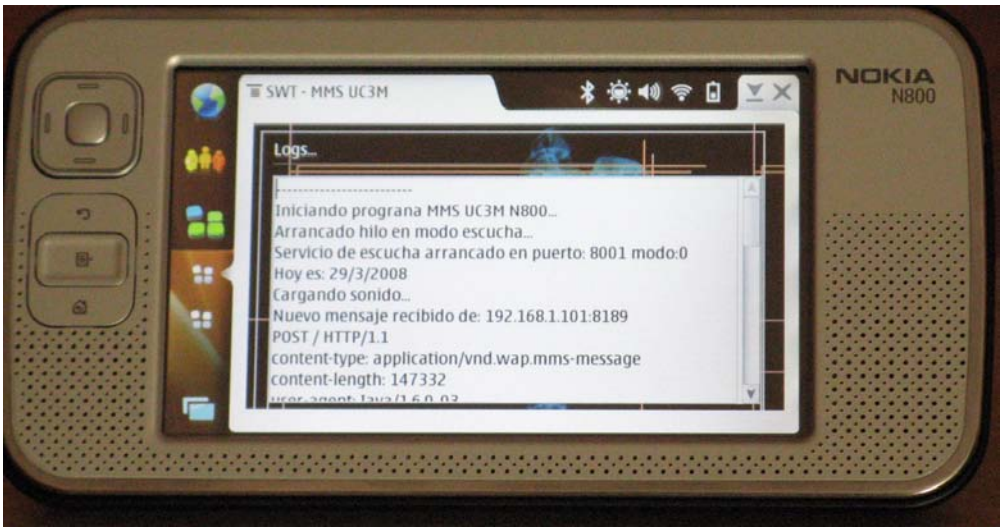
Si selecciona un archivo de sonido (wav) de la lista podrá oírlo al pinchar sobre él.

 Si es un mp3 o un video utilice el botón Ver videos, que abre el reproductor multimedia.

El botón Guardar almacenará el mensaje en la carpeta borradores para que pueda ser editado y enviado más tarde.

6.1.2.4. Logs

El botón de Logs, útil para desarrolladores, muestra en pantalla los problemas que hayan surgido al realizar una operación o la evolución de la aplicación (puertos en escucha, información de mensajes recibidos o enviados, etc…)



6.1.2.5. Carpeta Borradores

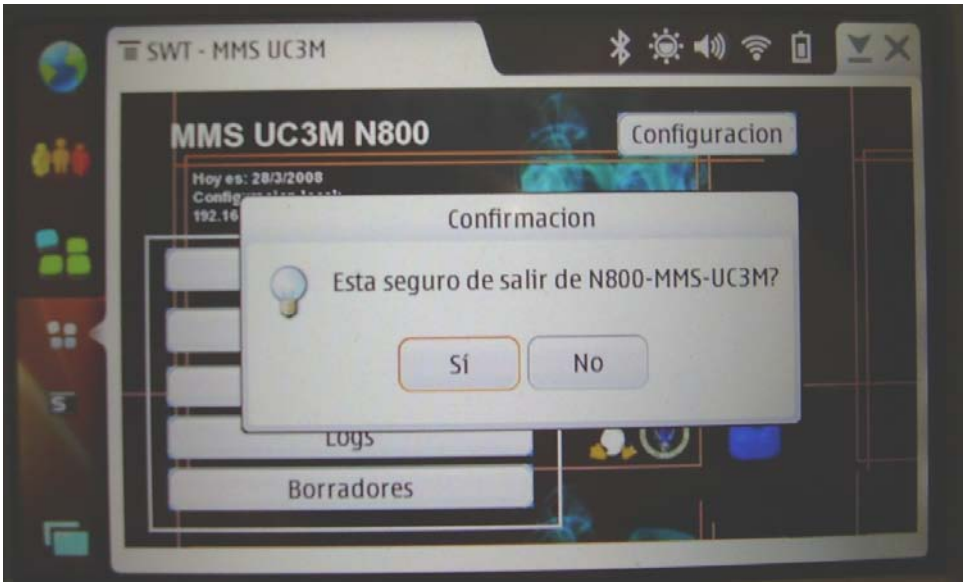
En esta carpeta se almacenarán los mensajes Nuevos que se guarden en lugar de enviarse. El funcionamiento es el mismo que en el caso de las otras dos bandejas.

6.1.2.6. Configuración

Con esta funcionalidad podremos cambiar el puerto de escucha para la recepción de mensajes o la URL donde deseamos enviar nuestro mensaje (MMSC u otro terminal N800 (IP más puerto de escucha)).

El campo IP MMSC puede ser utilizado para introducir la url del terminal al que se desea mandar el mensaje o el centro de mensajería global. Si se opta por esta segunda opción, será el centro de mensajería el que determine la IP del terminal al que corresponde el número de teléfono que se ponga en el campo Para de los mensajes a enviar.

Para abandonar cualquier pantalla e incluso la herramienta sólo pinche sobre la cruz de arriba a la derecha (Cerrar). Disfrute de la aplicación.



Información útil para desarrolladores. Cuando cierre la aplicación, los logs se descargarán al fichero /home/user/n800mms/log.dat donde podrá consultarlo desde

osso_xterm. Este fichero de logs es un histórico, por lo que puede llegar a alcanzar un gran tamaño, asegúrese de borrarlo o desplazarlo de carpeta cada cierto tiempo.

7. GESTIÓN DEL PROYECTO

La planificación inicial que se planteó para este proyecto fue cumplida en la medida de lo posible. A grandes rasgos, dicha planificación no sufrió modificaciones drásticas en el tiempo estimado para cada tarea. Salvo pequeños reajustes, propios del desarrollo de cualquier trabajo, las líneas maestras trazadas al inicio fueron cumplidas rigurosamente.

Se presenta en este apartado un resumen de las tareas realizadas y su consecución en el tiempo. Para ello se ha utilizado la herramienta gráfica conocida como Diagrama de Gantt, que permite mostrar el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado. Del mismo modo se justifican los costes del proyecto, tanto desde el punto de vista del material como de personal.

7.1. Planificación Final

A continuación se puede visualizar un diagrama de Gantt realizado con el programa ***GanttProject*** disponible para su descarga y uso bajo licencia GPL en la dirección oficial [32] . Desde sus inicios, estos diagramas se han convertido en una herramienta básica en la gestión de proyectos de todo tipo, con la finalidad de representar las diferentes fases, tareas y actividades programadas como parte de un proyecto o para mostrar una línea de tiempo en las diferentes actividades haciendo el método más eficiente.

[32] Sitio oficial ***GanttProject***: http://ganttproject.biz/.

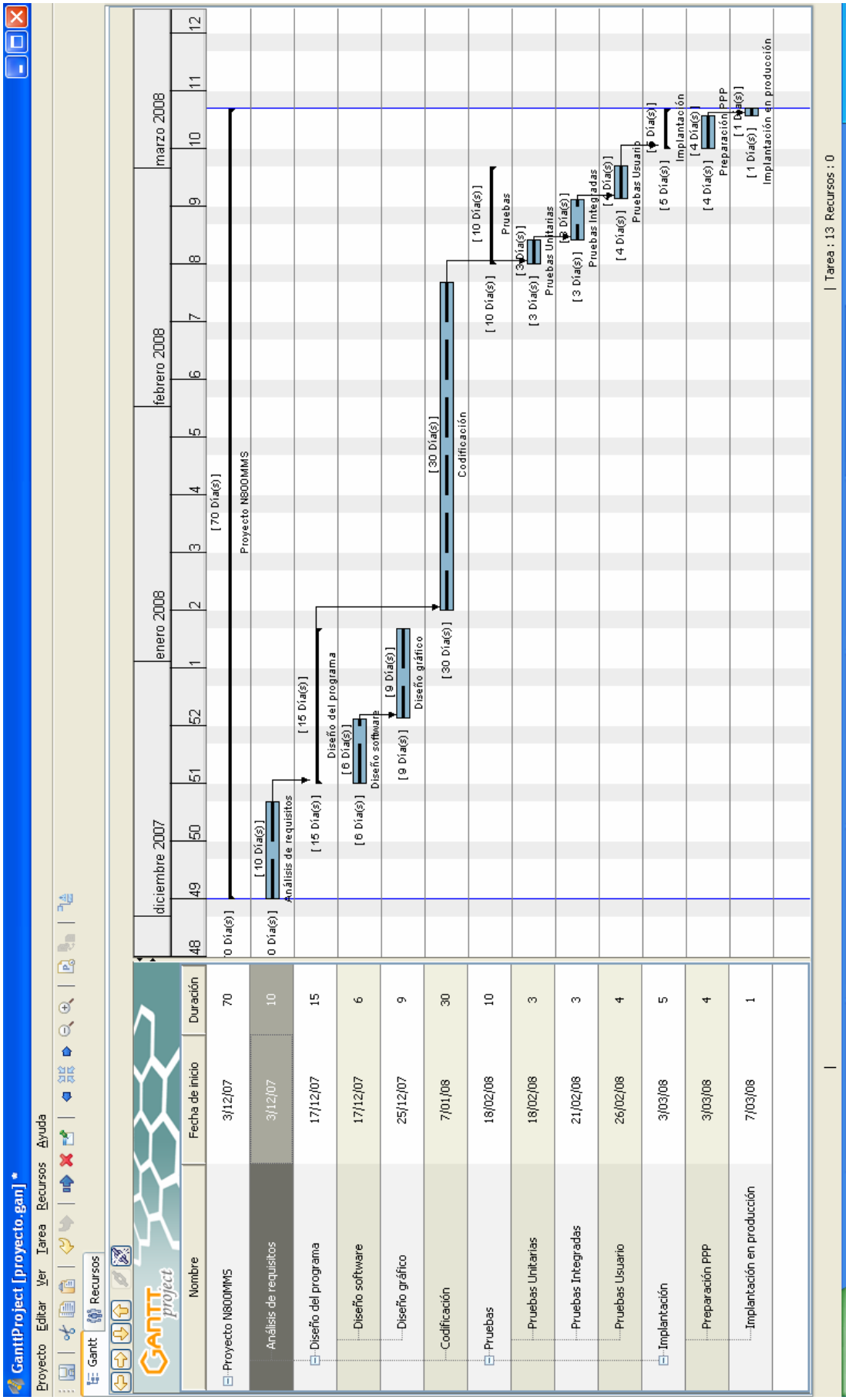


Ilustración 63 - Planificación del proyecto

7.2. Resumen de costes

Este proyecto comenzó a principios de diciembre del 2007 y finalizó el en marzo del año 2008, tal y como puede verse en el diagrama de Gantt presentado en el punto anterior, lo que era requisito indispensable del PFC puesto que la asignatura LIS se imparte en el segundo cuatrimestre y las prácticas comenzaban en abril. Las horas dedicadas a cada fase se repartieron de la siguiente manera:

Fase	Jornadas
Análisis de requisitos	10
Diseño del programa	15
Codificación	30
Pruebas	10
Preparación e Implantación	5

Tabla 22 - Resumen de jornadas por fase

Así pues se desprende que el número de horas dedicadas por el proyectando es de 70 jornadas. Dado que el coste por jornada para un desarrollador de aplicaciones se ha computado a 135 € / jornada (1 jornada 8 h), el coste de personal se sitúa en 9450 €.

Puesto que el coste de los terminales no es imputable a este proyecto pues la partida corre a cargo del cliente, el presupuesto total se resume en la siguiente tabla:

Concepto	Importe
Coste personal	9.450 €
Coste material	0 €
Base imponible	9.450 €
I.V.A. (16%)	1.512 €
Total	10.962 €

Tabla 23 - Resumen de costes

7.3. PLAN DE PUESTA EN PRODUCCIÓN

Identificación

Número	000001 – MMS_PUESTA_EN_PRODUCCIÓN
Fecha de Creación	13/07/2009

Datos generales

Nombre de la Implantación	000001 – MMS_PUESTA_EN_PRODUCCIÓN
Fecha propuesta de Implantación	19/07/2009
Departamento/Área solicitante	DIT/ LIS
Nueva Implantación	☒
Mantenimiento Evolutivo (Release)	☐
Mantenimiento Correctivo Ordinario	☐
Mantenimiento Correctivo Emergencia	☐

Descripción del cambio

Puesta en producción nuevo aplicativo

Motivo del cambio

A petición de las áreas usuarias

Equipo de Implantación

Técnico / Teléfono	Disponibilidad	Actividades durante el plan
DIT. U.CARLOSIII MªJesús Martínez	In situ	Tareas descritas PPP

Plan del Cambio

Paso	Dep	Duración	Responsable/Teléfono	Descripción
BACKUP				
0		10 min	MMV	Hacer copia de seguridad del terminal
PREVIOS				
1		X	MMV	Descagar utilidad osso_xterm http://maemo-hackers.org/wiki/ossoxterm/bora/it2007.install
2		X	MMV	Descargar el paquete de instalación de: http://tablets-dev.nokia.com/nokia_N800.php introduciendo el serial de nuestro terminal y bajándonos el paquete: RX-34_2008SE_2.2007.51-3_PR_COMBINED_MR0_ARM.bin 141226447 Software Edition 2008 version 2.2007.51-3 for Nokia N800 The second IT OS 2008 release for Nokia N800
3		X	MMV	Descargar de http://tablets-dev.nokia.com/d3.php flasher-3.0 40236 flasher utility (Linux on Intel x86) for the Nokia N800 and 770
4		X	MMV	Descargar de http://jalimo.evolvis.org/repository/maemo/jalimo-install.php la plataforma jalimo.
5		X	MMV	Crear el CD de Instalación
SUBIDA DESARROLLOS				
6	5	5 min	MMV	Instalar desde la carpeta Drivers del CD de instalación (llevándola desde el PC a la N800 vía USB) la utilidad osso_xterm.
7	6	1h	MMV	**Flasheado del terminal.** 1.- Activar el **modo píldora roja** para operar como administrador de la N800 siguiendo los siguientes pasos: a) Menú > Ajustes > Administrador de aplicaciones b) Menú arriba a la izquierda > Herramientas > Catálogo aplicaciones c) Nuevo catálogo > En Dirección web ponemos **matrix** y le damos a cancelar. Aparecerá una ventana que nos preguntará Red pill or blue pill?, le damos a Red pill. 2.- La N800 viene con la distribución Bora OS2007 instalada. Para que la aplicación pueda funcionar necesitamos flashear la N800 para conseguir la distribución OS2008 Maemo 4 Chinook. Para ello en el CD de instalación en la carpeta Drivers se

				encuentra la imagen del firmware (RX-34_2008SE_2.2007.51-3_PR_COMBINED_MR0_ARM), la copiamos en local en el PC. En la carpeta Drivers también se encuentra el flash 3.0[33] para Linux (en Intelx86), lo instalamos en nuestro PC en la misma carpeta que la imagen. 3.- Asegúrese de que la N800 está completamente cargada. Apague el terminal y, **apagado**, conéctelo al PC vía USB. 4.- Desde el PC ejecutamos como root (o como usuario normal si tiene privilegios para usar el puerto USB): ./flasher-3.0 -F RX-34_2008SE_2.2007.51-3_PR_COMBINED_MR0_ARM.bin -f -R Aparecerá el siguiente texto: PC: "Suitable USB device not found, waiting" is displayed on the console 5.- A continuación enchufe el cargador de la batería a la N800 o bien, enciéndala apretando el botón de encendido mientras mantiene apretado el botón Home. Desde el PC irá viendo los mensajes que va dejando la imagen mientras se carga en el terminal, finalmente, la N800 se reiniciará sola automáticamente. Enhorabuena, ya tiene instalada la distribución Chinook. Acuérdese de volver a activar el modo blue pill siguiendo el mismo procedimiento descrito en el apartado 1.
8	7	10 min	MMV	**Máquina virtual** En este punto ya tenemos el entorno preparado en el terminal, ahora necesitamos instalar la máquina virtual java para la N800. La máquina virtual para la N800 se llama cacao, del proyecto jalimo, y debemos instalarla directamente en la N800 pasándola desde el CD de instalación al terminal vía USB. Enhorabuena, ahora ya puede ejecutar cualquier aplicación java en su N800.
9	8	5 min	MMV	Ejecutar desde osso_xterm (también disponible en el CD): wget http://blog.2blocksaway.com/files/godmode.deb
10	9	5 min	MMV	Descargar vía USB a la N800 el paquete mmsn800.zip y ejecutar desde osso_xterm unzip mmsn800.zip
11	10	5 min	MMV	Ejecutar desde la carpeta donde hayamos extraído la aplicación:

				a) ./soyroot.sh b) ./instalarN800.sh
12	11	5min	MMV	En este punto la aplicación ya ha sido instalada en su N800. Para abrir la aplicación vaya a Menú (en la parte izquierda del terminal). Elija Extras y verá el acceso directo a la aplicación. Seleccione N800 MMS SWT y aguarde unos instantes mientras la aplicación se carga. Enhorabuena, ya ha instalado el servicio de MMS para la N800 desarrollado por la UC3M.
13	5	1h 45 min	MMV	Repetir en todos los terminales del laboratorio
PRUEBAS USUARIO				
14	12	30 min	DIT	Comprobar que la aplicación ha sido correctamente instalada en el terminal ejecutándola y probándola entre varios terminales.
FIN PPP				

Elaborar un Plan de Puesta en Producción (triple P o PPP en empresas) tan meticuloso tiene por objeto poder independizar el equipo de desarrollo del equipo de implantación de la aplicación. En este caso nos aseguramos que el equipo de implantación, el cliente (IT de la UC3M) o el estudiante de LIS, será capaz de repetir este proceso totalmente documentado.

7.4. Estado del proyecto

Actualmente el proyecto ha alcanzado la fase de madurez, y la aplicación se encuentra plenamente operativa en su primera versión estable.

8. CONCLUSIONES

A raíz de este proyecto, las prácticas docentes sobre la materia de mensajería multimedia de la asignatura de Ingeniería de Servicios cuentan con una herramienta de trabajo que, por un lado, se ajusta a los frameworks proporcionados en clase y, por otro lado, hace más atractivo el uso de los mismos para entender el protocolo de mensajería multimedia.

Es necesario subrayar el hecho de que se trata de una herramienta destinada a la docencia y, tanto su concepción como su diseño y desarrollo, se han enfocado para este fin, dejando muchas posibilidades de desarrollo abiertas, pero orientadas (programación MMSC, SMIL, etc). No obstante, más allá de ser una herramienta docente, el dotar a un dispositivo de mano como la N800 de capacidad de comunicación, no siendo un terminal móvil al uso (que opera con la red móvil de un operador) permite vislumbrar la idea de una red móvil basada en una infraestructura wireless. La emulación de una red móvil a través de una red wifi permitiría, entre otras aplicaciones, dotar de comunicaciones móviles a un espacio en el que la red de telefonía normal no llega o no puede llegar. En cuanto a la comunicación hablada, la solución para este tipo de dispositivos pasa por comunicarse mediante voIP.

En cuanto al desarrollo en sí del proyecto, en un principio se puso especial atención a la captura de los requisitos de la aplicación. Para ello se mantuvieron reuniones con el cliente y se valoraron todas las posibilidades a tenor de los requisitos que iban apareciendo. Finalmente se optó por la solución que aquí se presenta, ya que se consideró como la que más se ajusta a las peticiones del cliente.

No se han encontrado graves problemas durante el proceso de desarrollo que hayan supuesto un gran retraso en el plan de trabajo, tan sólo algunos propiciados por las dificultades propias del uso de frameworks y tecnologías con las que no había trabajado, como Maemo. Además del desconocimiento de la tecnología se unía el hecho de que, este entorno de desarrollo en las fechas en las que se desarrolló este proyecto, no tenía un foro de usuarios demasiado amplio por lo que la resolución de cualquier problema que surgiese con el entorno era más una cuestión de investigación personal que de búsqueda de ayuda en la red.

En líneas generales, con este proyecto he puesto en práctica muchos de los conocimientos adquiridos tanto en la universidad como en mi puesto de trabajo actual. Además, me ha permitido conocer e investigar otros entornos de trabajo con los que no estaba familiarizada y aplicar ese conocimiento a otros desarrollos.

9. BIBLIOGRAFÍA

[1] Android. 2009. Disponible en http://www.android.com/ consultado en Septiembre 2009.

[2] Beck, K., Extreme Programming Explained. 1999. Embrace Change, The XP Series. Addison Wesley.

[3] Ben Collins-Sussman, Brian W. Fitzpatrick and C. Michael Pilato . 2006. Version Control with Subversion. O'Reilly.

[4] Cacao. 2009. Cacao JVM homepage. Información disponible en http://www.cacaojvm.org , consultado en Septiembre del 2009.

[5] CUEVAS AGUSTÍN, G. 2003. Gestión del proceso software. Madrid: Centro de Estudios Ramón Areces, S.A.

[6] DAC. 2009. Prototipado de sistemas empotrados. Disponible en http://docencia.ac.upc.edu/EPSC/PSE/documentos/Trabajos/Archivo/Trabajo_PDM.pdf, consultado en septiembre del 2009.

[7] DIT Documentos. 2009. Disponible en http://www.it.uc3m.es/documentos/DIT_A4.pdf, consultado en septiembre del 2009.

[8] Gaona-Pérez. 2009. Java Platform Micro Edition (Java ME) Mensajes MMS y SMS. Disponible en http://www.sicuma.uma.es/sicuma/independientes/argentina08/Gaona-Perez/inicio.html, consultado en Septiembre del 2009.

[9] GNU ClassPath. 2009. Disponible en http://www.gnu.org/ , consultado en Septiembre del 2009.

[10] Jacobson, Ivar & Booch, Grady & Rumbaugh, James. 1999. El lenguaje unificado de modelado. Addison-Wesley.

[11] Jalimo. 2009. Jalimo – Proyect. Información disponible en http://www.jalimo.org consultada en Septiembre del 2009

[12] Juan González Gómez, 2002. El servicio SMS: Un enfoque práctico. Disponible en http://www.scribd.com/doc/14706145/El-servicio-SMS-Un-enfoque-practico?autodown=txt, consulado en Octubre 2009.

[13] Log4java. 2009. Disponible en: http://logging.apache.org/log4j/1.2/index.html , consultado en Octubre 2009.

[14] Maemo. 2009. Información sobre la plataforma disponible en www.maemo.org, consultado en septiembre 2009.

[15] MAVEN. 2009. Apache Maven Project. Disponible en http://maven.apache.org/guides/getting-started/index.html, consultado en Septiembre del 2009.

[16] Maven 2, a first glance. 2008. Disponible en http://www.techjava.de/topics/2008/09/maven-2-a-first-glance/ , consultado en Octubre del 2009.

[17] MavenMagic. 2009. Disponible en http://www.theserverside.com/tt/articles/article.tss?l=MavenMagic, consultado en Septiembre del 2009.

[18] MMS Center Application Development Guide. 2002. Disponible en http://www.nowsms.com/discus/messages/485/mmscenterappdevguide-15197.pdf , consultado en Octubre del 2009.

[19] MMSLibrary. 2009. Librería MMS para Java de Nokia. Manual disponible en http://www.it.uc3m.es/~kukielka/service_engineering/laboratory/related_material/Nokia%20MMS%20Java%20Library%20v1.1.pdf, consultado en Septiembre del 2009.

[20] Moblin. 2009. Información sobre la plataforma disponible en www.moblin.org consultado en septiembre 2009.

[21] Multimedia Messaging Service 1.3. 2009. Open Mobile Alliance. Disponible en http://www.openmobilealliance.org/Technical/release_program/mms_v1_3.aspx , consultado en Septiembre 2009.

[22] MYECLIPSE. 2009. MyEclipse Web Project Tutorial. Disponible en http://www.myeclipseide.com/documentation/quickstarts/webprojects/, consultado en Septiembre del 2009.

[23] NAUGHTON, PATRICK & SCHILDT, HERBERT. 1997. Java. Manual de referencia. McGraw-Hill.

[24] NIELSEN, J. 1994. Usability Inspection Methods. Wiley.

[25] Nokia MMS Java Library. 2009. Disponible en http://www.it.uc3m.es/~kukielka/service_engineering/laboratory/related_material/Nokia%20MMS%20Java%20Library%20v1.1.pdf, consultado en 2009.

[26] OpenMoko. 2009. Disponible en http://www.openmoko-spain.org/tiki-index.php , consultado en Septiembre 2009

[27] PhoneME. 2009.Disponible en: https://phoneme.dev.java.net/ , consultado en Septiembre 2009

[28] Sánchez Acosta, Alejandro. 2008. Las nuevas tenologías abiertas en dispositivos móviles. O'BRIEN, TIM & CASEY, JOHN & FOX, BRIAN & SNYDER, BRUCE & VAN ZYL, JASON. 2008. Maven: The definitive Guide. Sonatype.

[29] SUBVERSION. 2009. Subversion HomePage. Disponible en http://subversion.tigris.org, consultado en Septiembre 2009.

[30] SWT. 2009. Disponible en: http://www.eclipse.org/swt/ , consultado en Septiembre 2009.

[31] Telecomkh. 2009. Disponible en http://www.telecomkh.com/es/electronica/internet/telefonia-movil/noticias/productos-y-servicios/noticias/productos-y-servicios/noticias/productos-y-servicios/ip/hspa/3g/intel/linux/maemo/moblin/nokia/965, consultado en 2009.

[32] WAP-209. 2009. WAP-209-MMSEncapsulation-20010601-a, WAP Forum especificación, disponible en http://www.wapforum.org, consultado en 2009.

10. GLOSARIO

3GPP - Third-Generation Partnership Program

ADD - Architectural Design Document

API - Application Programming Interface

ARM - Advanced RISC Machines

ASCII - American Standard Code for Information Interchange

AWT - Abstract Window Toolkit

BSC - Base Station Controller

BTS - Base Transceiver Station

CBC - *Cell Broadcast* Center

CBS - *Cell Broadcast* Service

CDMA - Code Division Multiple Access

CLDC - Connected Limited Device Configuration

COM - Component Object Model

CPU - Central Processing Unit

CVS - Concurrent Versions System

DCM - Dispositivo de Cómputo Móvil

DDD - Detailed Design Document

DRM - Direct Rendering Manager

ECJ - Eclipse *Compiler* for Java

EIAF - Enterprise information Architecture Framework

EIR - Equipment Identity Register

FIC - First International Computer

GC - Garbage Collector

GEF - Graphic Editing Framework

GIMP - GNU Image Manipulation Program

GMSC - Gateway Mobile Services Switching Center

GNOME – GNU Network Object Model Environment

GNU - acrónimo recursivo que significa GNU No es Unix

GPL- General Public License

GPRS - General Packet Radio Service

GPS - Global Positioning System

GSM - Groupe Spécial Mobile

GTA - GNU Telephony Appliance

GTK+ - The GIMP Toolkit

GUI - Graphic User Interface

HLR - Home Location Register

HTTP - HyperText Transfer Protocol

HTTPS - HyperText Transfer Protocol Secure

IDE - Integrated Development Environment

IP - Internet Protocol

IVI - In-Vehicle Infotainment

J2SE - Standard Edition

J2EE - Enterprise Edition

J2ME - Java Micro Edition

JAR - Java Archive

JCP – Java Community Process

JDK - Java Development Kit

JDT - Java Development Toolkit

JIT - Just-In-Time

JRE - Java Runtime Environment

JSR - Java Specification Request

JVM - Java Virtual Machine

LIS - Laboratorio de Ingeniería del Software

MIC - Moblin Image Creator

MID – Mobile Internet Device

MIDP - Mobile Information Device Profile BS - Base Station

MIME - Multipurpouse Internet Mail Extensions

MMS - Multimedia Message System

MMSC - Multimedia Message System Center

MMSE - Multimedia Message System Environment

MMSNA - Multimedia Message System Network Architecture

MO-SM - Mobile Originated-Short Message

MP3 - MPEG-1 Audio Layer 3

MPEG - Moving Picture Experts Group

MS - Mobile Station

MSC - Mobile Services Switching Center

MSISDN - Mobile Station Integrated Services Digital Network

MT-SM - Mobile Terminated-Short Message

NDK - Native Development Kit

NSS - Network and Switching Subsystem

OMA - Open Mobile Alliance

OSA - Open Services Architecture

OSGi - Open Services Gateway Initiative

PDA - Personal Digital Assistant

PDU - Protocol Data Unit

PHD - Project History Document

PLMN - Public Land Mobile Network

POM - Proyect Object Model

RCP - Rich Client Platform

RFC - Request For Comments

RMI - Remote Method Invocation

RSS - Radio SubSystem

SDK - Software Development Kit

SIM - Subscriber Identity Module

SMS - Short Message Service

SMSC - Short Message Service Center

SOAP - Simple Object Access Protocol

SRD - Software Requirements Document

SSH - Secure SHell

SSL - Secure Sockets Layer -Protocol

SSOO - Sistemas Operativos

STD - Software Transfer Document

SUM - Software User Manual

SWT - Standard Widget Toolkit

TCP - Transmission Control Protocol

TDMA - Time Division Multiple Access

UAT - User Acceptance Testing

UML - Unified Modeling Language

URD - User Requirements Document

URL - Uniform Resource Locator

VAS – Value Added Service

VASA - Value Added Service Application

VASP - Value Added Service Provider

VLR - Visitor Location Register

VM - Virtual Machine

W3C - World Wide Web Consortium

WAP – Wireless Application Protocol

WLAN - Wireless Local Area Network

WMA - Wireless Messaging API

WSDL - Web Services Description Languaje

WTLS - Wireless Transmission Layer Security

XML - Extensible Markup Language

11. APÉNDICES

11.1. UML (Unified Modeling Language)

11.1.1. DIAGRAMA DE CASOS DE USO

La técnica de casos de uso, desarrollada por Ivar Jacobson, tiene como objetivo identificar los requisitos funcionales de un sistema estructurados en torno a las diversas categorías de usuarios [10]. El modelo de casos de uso describe cómo un actor usa un sistema para conseguir un objetivo y lo que el sistema hace para ayudarle. Así pues, nos sirve para definir y expresar gráficamente el sistema que nos concierne y su entorno. Gráficamente nos encontraremos con tres elementos claramente diferenciados:

- **Casos de uso**: son las funcionalidades que contiene el sistema. Se puede definir también como una colección de escenarios (entendiendo por escenario a la secuencia de acciones del actor y las acciones del sistema que describe una interacción típica entre ambos) con un objetivo común.

- **Actores**: son los agentes externos que interaccionan con el sistema. Los actores significan roles, no entidades concretas, y más que un rol, un conjunto coherente de roles, dado que un mismo actor puede participar en varios casos de uso desempeñando un rol diferente en cada uno de ellos.

- **Asociaciones**: son las relaciones entre agentes externos y funcionalidades.

Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema, al mostrar la reacción producida como respuesta a los eventos que se producen en el mismo. En la siguiente figura se muestra un ejemplo.

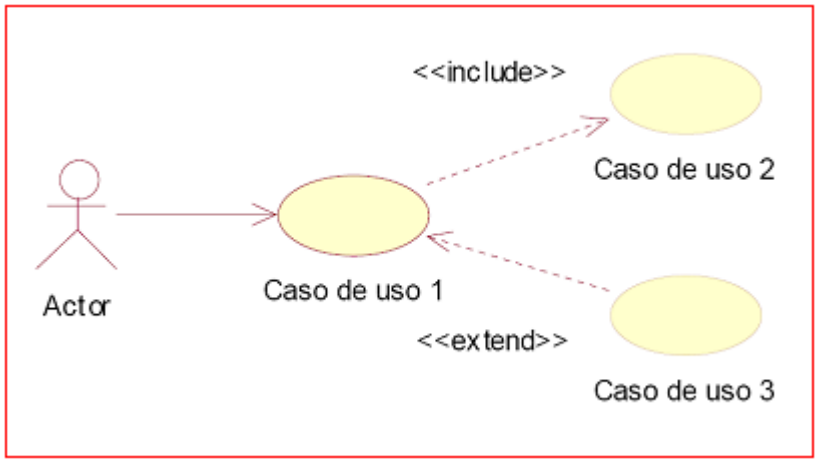


Ilustración 64 - Ejemplo de casos de uso

11.1.2. DIAGRAMA DE CLASES

Los diagramas de clases [10] son los más utilizados en el modelado de sistemas orientados a objetos. Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones.

Los diagramas de clases se utilizan para modelar la vista de diseño estática de un sistema. Principalmente, esto incluye modelar el vocabulario del sistema, modelar las colaboraciones y modelar esquemas. Los diagramas de clases también son la base para un par de diagramas relacionados: los diagramas de componentes y los de despliegue.

Algunos de los elementos que se pueden clasificar como estáticos en UML, son los siguientes:

Paquete Es el mecanismo de que dispone UML para organizar sus elementos en grupos.

Clase Describe un conjunto de objetos que comparte los mismos atributos, operaciones, métodos, relaciones y significado. Los componentes de una clase son:

Atributo: Se corresponde con las propiedades de una clase.

Operación: También conocido como método, es un servicio proporcionado por la clase que puede ser solicitado por otras clases y que produce un comportamiento en ella cuando se realiza.

Ilustración 65 - Representación gráfica de una clase

Interfaz: Representa a la funcionalidad que proporciona uno o varios elementos del modelo al resto de elementos. Esta interfaz no tiene asociado un comportamiento, por lo que tendrá que ser implementado por otro elemento. En estos diagramas se muestran, además, las relaciones estructurales existentes entre los elementos descritos anteriormente.

11.1.3. DIAGRAMAS DE COMPORTAMIENTO

Estos diagramas muestran la dinámica del sistema, por lo que complementan a los anteriores, que son estáticos.

11.1.3.1. Diagrama de estados

Representan la secuencia de los posibles estados por los que un objeto, o una interacción entre objetos, transcurre durante su tiempo de vida, en respuesta a los eventos recibidos. Modela el sistema como una máquina de estados. Un estado en

UML se puede definir como la situación en que se encuentra un objeto o una interacción cuando satisface una condición, desarrolla alguna acción o se encuentra esperando un evento. Cuando un objeto o una interacción pasa de un estado a otro, debido a un evento, se dice que ha sufrido una transición. Puede suceder que, con la aparición de un evento, el estado de origen y el de destino sea el mismo, teniendo así una transición reflexiva.

11.1.3.2. Diagramas de actividades.

Son similares a los diagramas de flujo tradicionales. En realidad, se corresponden con un caso especial de los diagramas de estado, donde los estados son estados de acción, y las transiciones vienen provocadas por la finalización de las acciones que tienen lugar en los estados de origen. Es decir, estados con una acción interna y una o más transiciones que

suceden al finalizar esta acción, o dicho de otro modo, un paso en la ejecución de lo que será un procedimiento.

Generalmente, estos diagramas están asociados a la implementación de un caso de uso, o de un método. Los diagramas de actividad se utilizan para mostrar el flujo de operaciones que se desencadenan en un procedimiento interno del sistema.

11.1.4. DIAGRAMAS DE INTERACCIÓN

Estos diagramas muestran las relaciones de interacción que tienen los objetos entre sí. También tienen cierto carácter dinámico, ya que representan las interacciones a lo largo del tiempo.

Los dos tipos de estos diagramas, secuencia y colaboración, son equivalentes. La diferencia está en que en el diagrama de secuencia se pone más énfasis en el transcurso del tiempo, mientras que en el diagrama de colaboración se enfatizan más las relaciones de interacción.

11.1.4.1. Diagramas de secuencia.

Muestran las interacciones entre un conjunto de objetos, ordenadas según el momento del tiempo en que tienen lugar. El objeto puede existir sólo durante la ejecución de la interacción, se puede crear o puede ser destruido durante la ejecución de la interacción. Un diagrama de secuencia representa una forma de indicar el periodo durante el que un objeto está desarrollando una acción directamente o a través de un procedimiento.

En este tipo de diagramas también intervienen los mensajes, que son la forma en que se comunican los objetos. Los mensajes consisten en la solicitud, por parte de un objeto origen, de una operación de un objeto destino. Existen distintos tipos de mensajes según cómo se producen en el tiempo: simples, síncronos, y asíncronos.

Los diagramas de secuencia permiten indicar cuál es el momento en el que se envía o se completa un mensaje mediante el tiempo de transición, especificado en el diagrama.

11.1.4.2. Diagramas de colaboración.

Muestran la interacción entre varios objetos y los enlaces que existen entre ellos. Representa las interacciones entre objetos, organizadas alrededor de los objetos, y sus vinculaciones. A diferencia de un diagrama de secuencia, un diagrama de colaboraciones muestra las relaciones entre los objetos, no la secuencia en el tiempo en que se producen los mensajes. Los componentes de los diagramas de colaboración son los objetos, enlaces y

mensajes. Un enlace es una instancia de una asociación que conecta dos objetos de un diagrama de colaboración. El enlace puede ser reflexivo si conecta a un elemento consigo mismo. La existencia de un enlace entre dos objetos indica que puede existir un intercambio de mensajes entre los objetos conectados. Los diagramas de interacción muestran el flujo de mensajes entre elementos del modelo. El flujo de mensajes representa el envío de un mensaje desde un objeto a otro, si entre ellos existe un enlace.

Los mensajes que se envían entre objetos pueden ser de distintos tipos, también según como se produzcan en el tiempo. Los tipos de mensajes básicos son, al igual que en los diagramas de secuencia, simples, síncronos y asíncronos.

11.1.5. DIAGRAMAS DE IMPLEMENTACIÓN

A diferencia del resto de diagramas, que expresan la lógica del sistema, estos diagramas modelan el sistema desde un punto de vista físico. Se entiende como físico también, los ficheros ejecutables y cualquier otro tipo de archivo que intervenga en el sistema.

11.1.5.1. Diagramas de componentes.

Un componente es una pieza del sistema que se está representando.

Muestra la dependencia entre los distintos componentes de software, incluyendo componentes de código fuente, binario y ejecutable.

11.1.5.2. Diagramas de despliegue.

Los diagramas de despliegue [10] muestran la configuración de los componentes hardware, los procesos, los elementos de procesamiento en tiempo de ejecución y los objetos que existen en tiempo de ejecución. En este tipo de diagramas intervienen nodos, asociaciones de comunicación, componentes dentro de los nodos y objetos que se encuentran a su vez dentro de los componentes.

Un nodo es un objeto físico en tiempo de ejecución, es decir, una máquina que se compone habitualmente de, por lo menos, memoria y capacidad de procesamiento, a su vez puede estar formado por otros componentes.

11.2. HERRAMIENTAS PARA LA ELABORACIÓN DEL PROYECTO

11.2.1. REQUISITOS PREVIOS

Vamos a instalar el entorno de trabajo del proyecto sobre una distribución de Ubuntu. Si somos reacios a montar una partición en nuestro PC también podemos optar por la distribución Wubi (http://wubi-installer.org/) un instalador de Ubuntu para Windows, que monta Ubuntu como si fuese una carpeta en Windows. El arranque del PC es el mismo que si hubiésemos optado por instalar Ubuntu directamente en una partición. Encontraremos las instrucciones para la instalación en la página oficial de Ubuntu (http://www.ubuntu.com) o Wubi.

Una vez instalado el sistema operativo procederemos a instalar el JDK de java para poder desarrollar nuestro proyecto.

Como root del sistema instalamos java con el siguente comando: sudo apt-get install sun-java5-jdk

También descargamos el mp3plugin.jar o el jlayer1.jar para aplicarlo como librería a nuestro proyecto desde:

http://java.sun.com/javase/technologies/desktop/media/jmf/mp3/download.html.

11.2.2. ECLIPSE

Eclipse [22] es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).

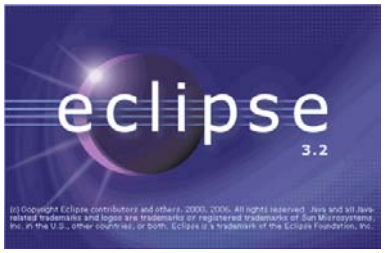


Ilustración 66 - Logo de Eclipse

Eclipse es también una comunidad de usuarios, extendiendo constantemente las áreas de aplicación cubiertas. Un ejemplo es el recientemente creado Eclipse Modeling Project, cubriendo casi todas las áreas de Model Driven Engineering.

Eclipse fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

La base para Eclipse es la Plataforma de cliente enriquecido (del Inglés Rich Client Platform RCP). Los siguientes componentes constituyen la plataforma de cliente enriquecido:

- **Plataforma principal** Inicio de Eclipse, ejecución de plugins.
- **OSGi** Una plataforma para bundling estándar.
- **Standard Widget Toolkit (SWT)** Un widget toolkit portable.
- **JFace** Manejo de archivos, manejo de texto, editores de texto
- **Workbench** Vistas, editores, perspectivas, asistentes

Los widgets de Eclipse están implementados por un herramienta de widget para Java llamada SWT, a diferencia de la mayoría de las aplicaciones Java, que usan las opciones estándar Abstract Window Toolkit (AWT) o Swing. La interfaz de usuario de Eclipse también tiene una capa GUI intermedia llamada JFace, la cual simplifica la construcción de aplicaciones basada en SWT.

El entorno de desarrollo integrado (IDE) de Eclipse emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad al frente de la plataforma de cliente rico, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software. Adicionalmente a permitirle a Eclipse extenderse usando otros lenguajes de programación como son C/C++ y Python, permite a Eclipse trabajar con lenguajes para procesado de texto como LaTeX, aplicaciones en red como Telnet y Sistema de gestión de base de datos. La arquitectura plugin permite escribir cualquier extensión deseada en el ambiente, como sería Gestión de la configuración. Se provee soporte para Java y CVS en

el SDK de Eclipse. Y no tiene por qué ser usado únicamente para soportar otros lenguajes de programación.

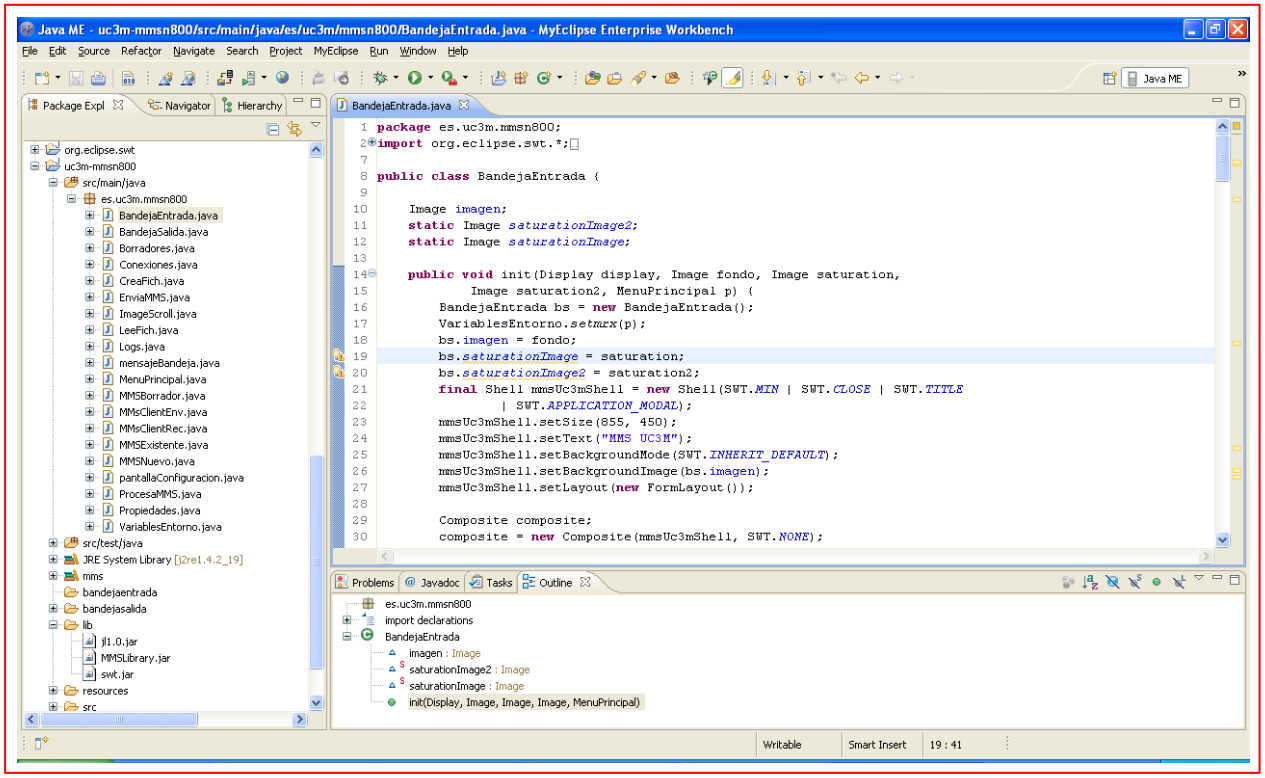


Ilustración 67 - Jerarquía de paquetes del proyecto MMS N800 en Eclipse.

La definición que da el proyecto Eclipse acerca de su software es: "una especie de herramienta universal - un IDE abierto y extensible para todo y nada en particular".

En cuanto a las aplicaciones clientes, eclipse provee al programador con frameworks muy ricos para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones web, etc. Por ejemplo, GEF (Graphic Editing Framework - Framework para la edición gráfica) es un plugin de Eclipse para el desarrollo de editores visuales que pueden ir desde procesadores de texto hasta editores de diagramas UML, interfaces gráficas para el usuario (GUI), etc. Dado que los editores realizados con GEF "viven" dentro de Eclipse, además de poder ser usados conjuntamente con otros plugins, hacen uso de su interfaz gráfica personalizable y profesional.

El SDK de Eclipse incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código. El IDE también hace uso de un espacio de trabajo, en este caso un grupo de metadata en un espacio para archivos plano,

permitiendo modificaciones externas a los archivos en tanto se refresque el espacio de trabajo correspondiente.

La versión actual de Eclipse dispone, entre otras, de las siguientes características: editor de texto, resaltado de sintaxis, compilación en tiempo real, pruebas unitarias con JUnit, control de versiones con CVS, integración con Ant, asistentes (wizards) para la creación de proyectos, clases, tests, etc., refactorización de clases, métodos, etc., gestión de plugins y muchas otras.

11.2.2.1. Instalación de Eclipse

Para instalar Eclipse basta descargar el fichero comprimido de la página oficial de Eclipse (http://www.eclipse.org/) y descomprimirlo en la carpeta que queramos, no precisa instalación.

Si en nuestro sistema tenemos más de una jdk instalada de la que tiran otros programas podemos independizar el arranque de nuestro eclipse de las variables de sistema. Para ello podemos utilizar el siguiente acceso directo a nuestro eclipse (guardar las siguientes líneas como eclipse.bat):

```
set JAVA_HOME=”camino a la jdk”
set MVN_BIN=”camino a maven”\apache-maven-2.2.1
PATH=%JAVA_HOME%\jre;%JAVA_HOME%\jre\bin;%JAVA_HOME%\jre\bin\client;%MVN_BIN%\bin
cd “camino a eclipse”\eclipse\
@eclipse.exe
```

Tabla 24 - Script de ejecución de eclipse

Para el desarrollo de este proyecto se utilizó la versión Ganymede.

11.2.3. SUBVERSIÓN

Para el desarrollo de este proyecto se ha utilizado un sistema de control de versiones muy práctico que ha permitido poder desarrollar desde distintas ubicaciones y poder volver atrás en determinados momentos del desarrollo.

Subversion [29] es un sistema de control de versiones libre y de código fuente abierto. Es decir, Subversion maneja ficheros y directorios a través del tiempo. Hay un árbol de ficheros en un repositorio central. El repositorio es como un servidor de ficheros ordinario, excepto porque recuerda todos los cambios hechos a sus ficheros y directorios. Esto le permite recuperar versiones antiguas de sus datos, o examinar el historial de cambios de los mismos.

En este aspecto, mucha gente piensa en los sistemas de versiones como en una especie de “máquina del tiempo”.

Subversion [3] puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. A cierto nivel, la capacidad para que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Se puede progresar más rápidamente sin un único conducto por el cual deban pasar todas las modificaciones. Y puesto que el trabajo se encuentra bajo el control de versiones, no hay razón para temer por que la calidad del mismo vaya a verse afectada por la pérdida de ese conducto único si se ha hecho un cambio incorrecto a los datos, simplemente se deshace ese cambio.

Algunos sistemas de control de versiones son también sistemas de administración de configuración de software. Estos sistemas son diseñados específicamente para la administración de árboles de código fuente, y tienen muchas características que son específicas del desarrollo de software, tales como el entendimiento nativo de lenguajes de programación, o el suministro de herramientas para la construcción de software. Sin embargo, Subversion no es uno de estos sistemas. Subversion es un sistema general que puede ser usado para administrar cualquier conjunto de ficheros. Para usted, esos ficheros pueden ser código fuente, para otros, cualquier cosa desde la lista de la compra de comestibles hasta combinaciones de vídeo digital y más allá.

Ilustración 68 - Logo de Subversión

Características de Subversion

Al discutir acerca de las características que Subversion aporta al mundo del control de versiones, a menudo es útil hablar de ellas en términos de cómo han mejorado el diseño de CVS (Concurrent Versions System), otro sistema de control de versiones anterior a Subversion. Vamos a ver un listado de esas mejoras:

- **Versionado de directorios** CVS solamente lleva el historial de ficheros individuales, pero Subversion implementa un sistema de ficheros versionado "virtual" que sigue los cambios sobre árboles de directorios completos a través del tiempo. Ambos, ficheros y directorios, se encuentran bajo el control de versiones.

- **Verdadero historial de versiones** Dado que CVS está limitado al versionado de ficheros, operaciones como copiar y renombrar, las cuales pueden ocurrir sobre ficheros, pero que realmente son cambios al contenido del directorio en el que se encuentran, no son soportadas por CVS. Adicionalmente, en CVS no se puede reemplazar un fichero versionado con algo nuevo que lleve el mismo nombre sin que el nuevo elemento herede el historial del fichero antiguo, que quizás sea completamente distinto al anterior. Con Subversion, usted puede añadir, borrar, copiar, y renombrar ficheros y directorios. Y cada fichero nuevo añadido comienza con un historial nuevo, limpio y completamente suyo.

- **Envíos atómicos** Una colección cualquiera de modificaciones o bien entra por completo al repositorio, o bien no lo hace en absoluto. Esto permite a los desarrolladores construir y enviar los cambios como fragmentos lógicos e impide que ocurran problemas cuando sólo una parte de los cambios enviados lo hace con éxito.

- **Versionado de metadatos** Cada fichero y directorio tiene un conjunto de propiedades, claves y sus valores, asociado a él. Usted puede crear y almacenar cualquier par arbitrario de clave/valor que desee. Las propiedades son versionadas a través del tiempo, al igual que el contenido de los ficheros.

- **Elección de las capas de red** Subversion tiene una noción abstracta del acceso al repositorio, facilitando a las personas implementar nuevos mecanismos de red. Subversion puede conectarse al servidor HTTP Apache como un módulo de extensión. Esto proporciona a Subversion una gran ventaja en estabilidad e

interoperabilidad, y acceso instantáneo a las características existentes que ofrece este servidor-autenticación, autorización, compresión de la conexión, etcétera. También tiene disponible un servidor de Subversion independiente, y más ligero. Este servidor habla un protocolo propio, el cual puede ser encaminado fácilmente a través de un túnel SSH (Secure SHell).

- **Manipulación consistente de datos** Subversion expresa las diferencias del fichero usando un algoritmo de diferenciación binario, que funciona idénticamente con ficheros de texto (legibles para humanos) y ficheros binarios (ilegibles para humanos). Ambos tipos de ficheros son almacenados igualmente comprimidos en el repositorio, y las diferencias son transmitidas en ambas direcciones a través de la red. Ramificación y etiquetado eficientes. El coste de ramificación y etiquetado no necesita ser proporcional al tamaño del proyecto. Subversion crea ramas y etiquetas simplemente copiando el proyecto, usando un mecanismo similar al enlace duro. De este modo estas operaciones toman solamente una cantidad de tiempo pequeña y constante.

- **Hackability** Subversion no tiene un equipaje histórico; está implementado como una colección de bibliotecas compartidas en C con APIs bien definidas. Esto hace a Subversion extremadamente fácil de mantener y reutilizable por otras aplicaciones y lenguajes.

Es por todas estas cualidades por las que se ha decidido utilizar subversion como herramienta de gestión de versiones. Para facilitar aún más el control del repositorio, se ha utilizado la herramienta TortoiseSVN, que integra la funcionalidad de Subversion en el explorar de windows.

11.2.3.1. Instalación de subversión

Para instalar el sistema de control de versiones subversion debemos bajárnoslo de la página oficial de subversion (http://subversion.tigris.org/). Lo descargamos y seguimos la guía de instalación para establecer nuestro servidor de subversion.

Una vez instalado el servidor procederemos a descargarnos el plugin para eclipse de subversion: **subclipse** que nos será de gran utilidad para trabajar con nuestro proyecto y el control de versiones desde nuestro escritorio de trabajo de Eclipse de una forma más sencilla que con el **TortoiseSVN** que es otra opción. Ambos están disponibles en la página oficial del SVN.

Para instalar el plugin de Subversion para Eclipse, abrimos el eclipse y estando ya dentro del IDE hacemos:

click en Help->Install new software–>click en el botón add

Ahí colocamos lo siguiente:

Name: Subversion

Location: `http://subclipse.tigris.org/update_1.6.x` y damos click en ok.

Después nos aparecerá en pantalla con los paquetes que tenemos que instalar para poder hacer uso de este plugin. Seguimos las instrucciones y tras unos minutos y el reinicio del eclipse, tendremos instalado nuestro plugin. Para comprobarlo basta con ir a **Window-> Open perspective->Other**. En este menú deberá aparecernos lo que se muestra en la siguiente figura:

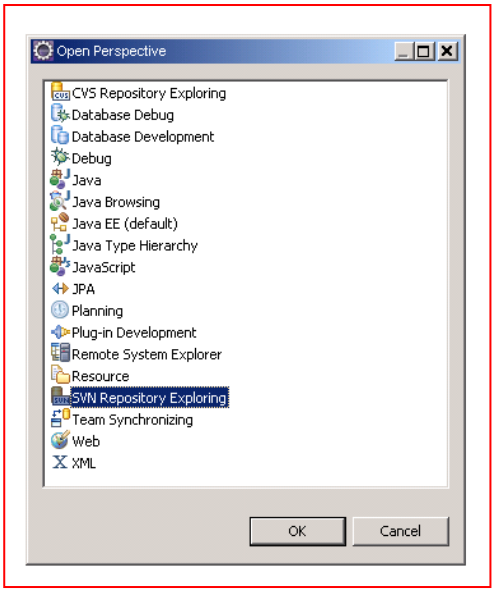


Ilustración 69 - Subclipse instalado

11.2.4. MAVEN

Maven [15] es una herramienta software para la creación y gestión de proyectos Java basado en el concepto Proyect Object Model (POM).

Características:

- Hacer el proceso de desarrollo sencillo.
- Proporcionar un sistema de desarrollo uniforme.

- Proporcionar información útil sobre el proyecto: archivo log, referencias cruzadas, listas de correo, listas de dependencias, reportes de tests unitarios.
- Proporcionar líneas básicas para la buena programación, por ejemplo, a la hora de especificar, ejecutar y documentar tests unitarios.
- Permitir actualizaciones sencillas a nuevas versiones.
- Gestión de dependencias.
- Repositorios remotos.
- Reutilización de la lógica.
- Integración en IDEs más importantes.

11.2.4.1. Filosofía de Maven

Mucha gente [28] compara Ant con Maven, sin embargo Ant es realmente una caja de herramientas mientras que Maven trata de aplicar una serie de patrones para conseguir una infraestructura de proyecto que cumpla unas características de visibilidad, reusabilidad y mantenimiento adecuadas. Sin estas características es muy improbable que varios programadores trabajen juntos de manera productiva en un proyecto. Sin visibilidad es difícil que un programador sepa lo que otros han programado ya, con la consecuencia de que ese código útil no sea reutilizado. Esto nos lleva al denominado efecto silo: el conocimiento compartido por los programadores es cada vez menor, mientras que la frustración de los mismos es cada vez mayor.

Ilustración 70 - Logo de Maven

Maven nace con el deseo de hacer multitud de proyectos siguiendo siempre el mismo camino. En un nivel muy abstracto todos los proyectos necesitan ser construidos, testeados, empaquetados, documentados y desplegados. Por supuesto que hay infinitas variaciones en el orden en que estos pasos son ejecutados, pero estas variaciones ocurren entre los límites de un camino bien definido y este es, precisamente, el camino que Maven trata de presentar al

mundo de manera clara y sencilla. La manera más sencilla de hacer un camino claro es proporcionar a la gente una serie de patrones que puedan ser compartidos por cualquiera que esté implicado en el proyecto.

Fases del ciclo de vida de un proyecto generado con Maven:

- Validación: el proyecto es correcto y la información disponible.
- Compilación: compilación del código del proyecto.
- Test: testear el código compilado. No es necesario que la aplicación esté desplegada.
- Empaquetado: generación, por ejemplo, de JARs.
- Integración de test: despliegue del código en un entorno donde los test puedan llevarse a cabo.
- Verificación: comprobación de que el paquete es correcto.
- Instalación: instalación del paquete en el repositorio local, para que pueda ser usado como dependencia de otros proyectos distintos.
- Despliegue.

Maven adopta un punto de vista distinto a Ant, ya que fuerza a trabajar de tal modo que se desarrollen módulos, conocidos como artifacts, que dependen de versiones concretas de librerías, que a su vez se encuentran en un repositorio común. Estos artifacts serán publicados en el repositorio, de manera que estarán disponibles para el resto de los desarrolladores del proyecto.

El primer concepto fundamental de Maven es el Project Object Model, plasmada en el archivo de configuración POM.xml. Este fichero contiene toda la información acerca del proyecto: nombre, tipo, versión, autor/autores, dependencias, etc.

Otro concepto clave de la filosofía Maven es el repositorio, lugar donde se guardan los artifacts de los cuales dependa el proyecto. La idea es que a través de este repositorio, local o remoto, los desarrolladores compartan todas las librerías.

La versatilidad de Maven reside en la cantidad de plugins que tiene disponible, que permiten hacer casi cualquier cosa. Además de estos plugins de serie pueden encontrarse plugins de terceros y si con eso no es suficiente, se pueden crear nuevos plugins que den solución al problema en particular.

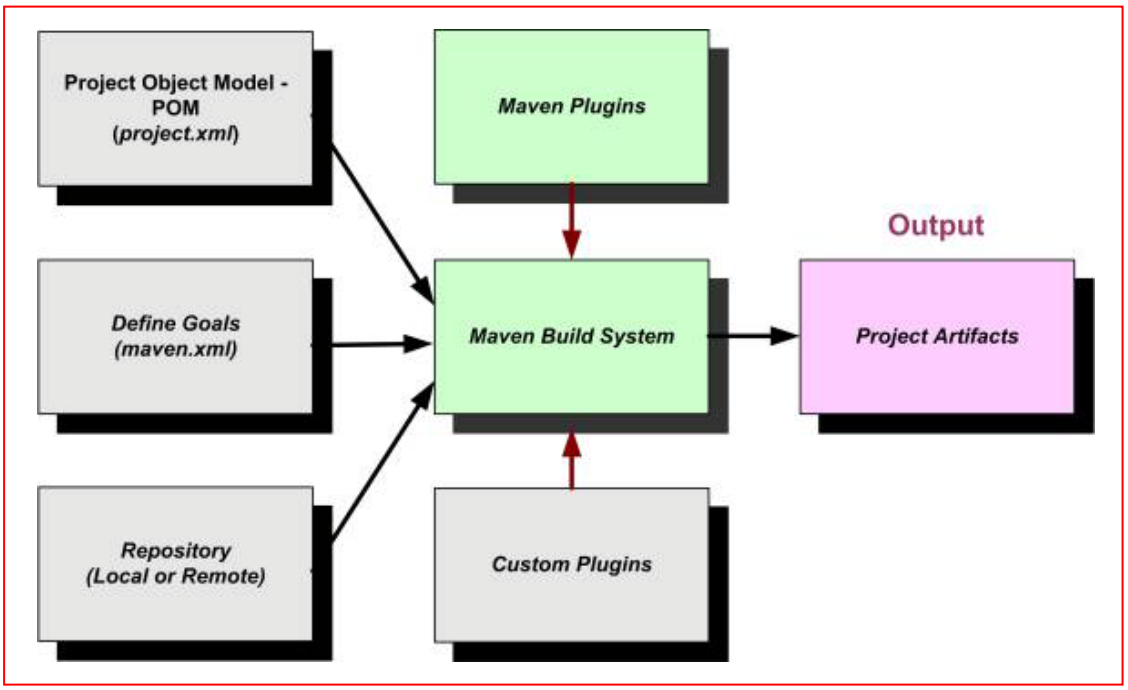


Ilustración 71 - Estructura funcional de Maven [17]

Los cuadros grises son proporcionados por el programador, los verdes por Maven y el rosa es el resultado final del proyecto.

Maven está claramente orientado al trabajo en equipo y consigue que:

1. El problema de las dependencias entre proyectos desaparezca.
2. Un mismo proyecto funcione en distintos entornos de desarrollo y/o sistemas operativos sin tocar el código.
3. El trabajo de un desarrollador se integre de forma transparente con el resto.
4. Se maximice la cohesión del código.
5. Se minimice el acoplamiento del código.
6. Se facilite la reutilización de código.

11.2.4.2. Maven 2.0

Los objetivos perseguidos por esta nueva versión han sido hacerlo más rápido, fácil de usar e implementar nuevas funciones de las que maven 1.x carece, como, por ejemplo, las dependencias transitivas.

Características de Maven 2.0:

Más rápido y más pequeño El núcleo de Maven ya no usa ni Ant, ni Jelly ni Xerces, lo que ha hecho que gane en rapidez y pierda tamaño.

Ciclo de construcción definido El proceso de construcción se estandariza en una serie de pasos bien definidos.

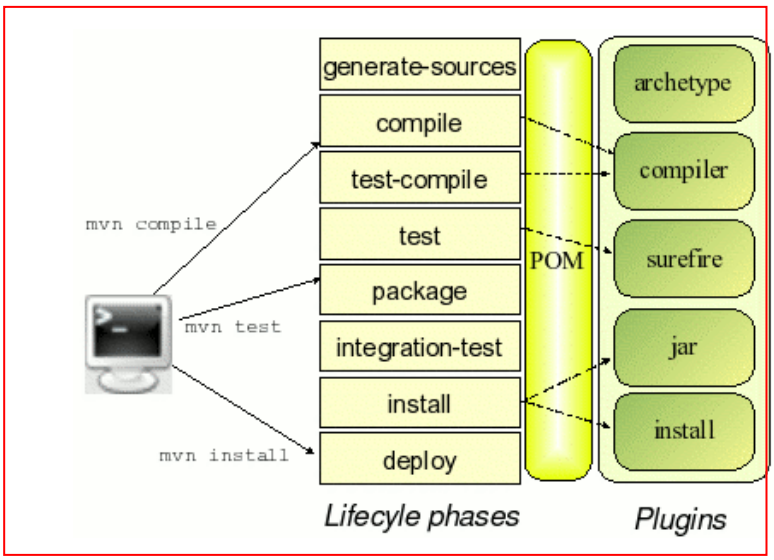


Ilustración 72 - Fases en Maven 2.0 (Maven2 first glance, 2008)

No se usa Jelly para desarrollar plugins Se escriben en Java.

Desaparece el fichero maven.xml La parametrización de los builds se hace a través de plugins.

Se ha mejorado la estructura del repositorio Más fácil navegar por él.

El fichero project.xml pasa a llamarse pom.xml.

Mejor soporte de snapshots Los snapshots son las últimas versiones de desarrollo disponibles de un proyecto Maven. Ahora por defecto se comprueban si existen actualizaciones una vez al día aunque puede cambiarse a cada vez que se haga un build, cada cierto intervalo de tiempo o nunca. También es posible forzar la comprobación de actualizaciones vía una opción en línea de comandos, de tal modo que sea muy similar al uso de un sistema de control de versiones.

11.2.4.3. Instalación de maven

Maven es operable a través de terminal o ventana de comandos. Lo podemos descargar de la página oficial (http://maven.apache.org/). Maven necesita de una jdk 5 o superior.

Primero debemos bajarnos la herramienta e instalarla en el entorno de trabajo. El enlace: http://maven.apache.org/download.html. Una vez descargado es necesario incluir su directorio bin en el path de búsqueda de ejecutables34.

Para comprobar que hemos incluido bien el path, podemos ejecutar mvn –v o.

Es posible que la primera vez que lo ejecutemos tarde un rato ya que cada vez que ejecutamos un comando nuevo de maven tiene que bajarse cosas de internet.

Una vez instalado Maven, la mayor parte del tiempo estaremos buscando ayuda sobre un plugin, para ver la información detallada de un plugin basta con ejecutar el siguiente comando:

mvn help:describe –Dplugin=help –Dfull

Este comando descargará la información necesaria de internet y tras obtenerla mostrará la descripción del plugin.

Al igual que con subversion, vamos a emplear el plugin de maven para eclipse en el desarrollo (http://maven.apache.org/eclipse-plugin.html). Para poder utilizarlo basta con haber instalado maven en nuestro sistema.

Para instalar el plugin de Maven para Eclipse, abrimos el eclipse y estando ya dentro del IDE hacemos:

click en Help->Install new software–>click en el botón add

Ahí colocamos lo siguiente:

Name: Maven 2

Location: http://m2eclipse.sonatype.org/update/

y damos click en ok.

[34] Para meter el subdirectorio **bin** en el path de ejecutables, botón derecho del ratón sobre el icono de "Mi PC" y elegimos "Propiedades", "Avanzado" y "Variables de Entorno". Buscamos "Path" y le damos a "Editar". Añadimos al final ";”camino a maven”\apache-maven-2.2.1 \bin" (con punto y coma delante, para separarlo de lo que ya haya escrito).

Después nos aparecerá en pantalla con los paquetes que tenemos que instalar para poder hacer uso de este plugin.

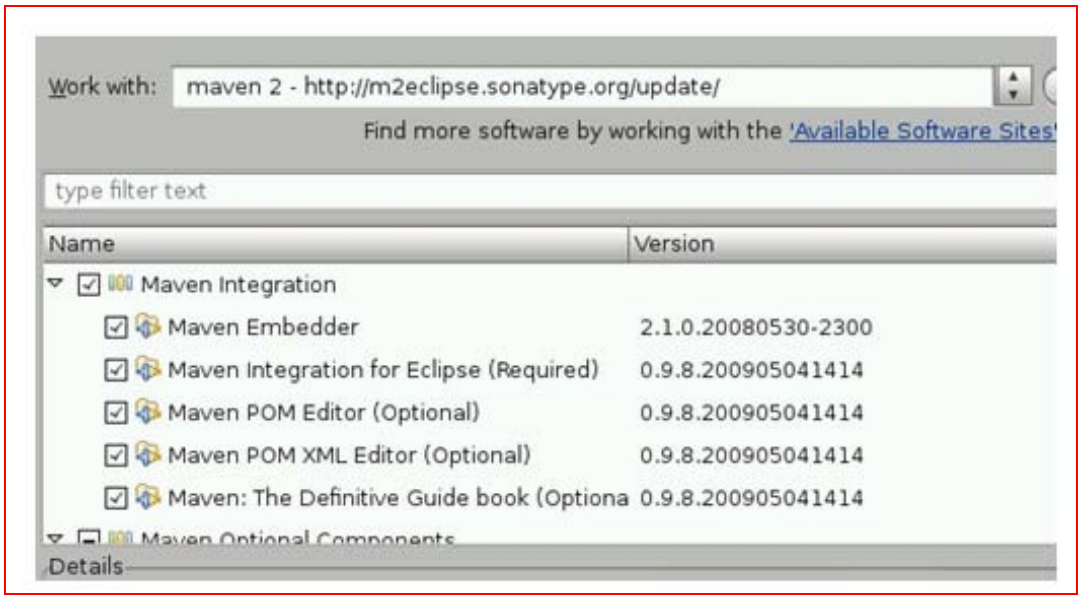


Ilustración 73 - Paquetes Maven para eclipse

De la lista que se desplegó elegiremos los siguientes:

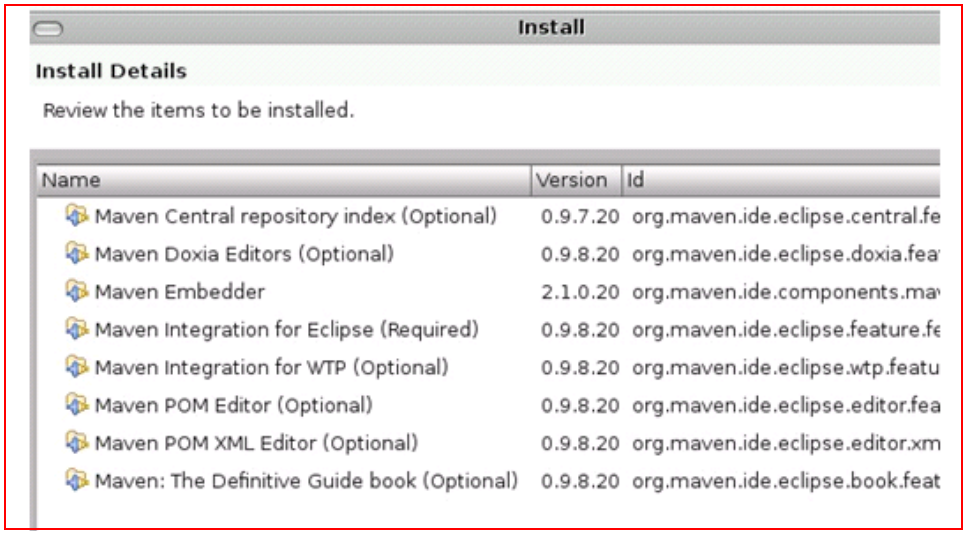


Ilustración 74 - Paquetes a seleccionar en la instalación de maven para eclipse

Ahora hacemos **click en el botón next y finish**. Por ultimo esperamos a que se descargue e instale el plugin. Una vez que se termino la instalación eclipse te preguntara que si deseas reiniciar el IDE así que por favor da **click en ok.**

Después de que eclipse haya reiniciado, en automático eclipse realiza una configuración de las propiedades Maven de nuestra maquina (por eso es necesario tener

previamente instalado Maven). Para verificar esto nos vamos **window–>preferences** y del lado izquierdo del árbol click en la opción **Maven–>installations** para ver las opciones de maven.

Con estos sencillos pasos podremos hacer uso de una forma visual de Maven.

11.2.5. MAEMO

El entorno de desarrollo para el proyecto que nos proporcionará la funcionalidad del sistema operativo real del dispositivo será Maemo. Puesto que en el apartado dedicado a Maemo en el capítulo del Estado del Arte ya hemos documentado los pormenores de este entorno de desarrollo, en este capítulo vamos a proceder a indicar cómo configurar el entorno de desarrollo para poder trabajar con maemo.

11.2.5.1. Instalación Maemo

Una vez instalado el sistema operativo procederemos a instalar el entorno de desarrollo siguiendo los siguientes pasos:

1. Descargamos la distribución deseada de maemo (en el desarrollo del proyecto hemos utilizado la distribución 4 Chinook) desde la página oficial de Maemo (http://maemo.org/intro/) o bien:

```
$ wget http://repository.maemo.org/stable/4.1.2/maemo-scratchbox-install_4.1.2.sh http://repository.maemo.org/stable/4.1.2/maemo-sdk-install_4.1.2.sh
```

2. Como root ejecuto:

```
$ echo 0 > /proc/sys/vm/vdso_enabled
```

3. Damos permiso a los ficheros descargados

```
$ chmod a+x ./maemo-scratchbox-install_4.1.2.sh
$ sudo ./maemo-scratchbox-install_4.1.2.sh
```

4. Añadimos nuestro usuario al grupo de trabajo de maemo

```
$ sudo ./sbox_adduser aurelio
```

Otra opción es ejecutar el script con la opción:

```
$ sudo ./maemo-scratchbox-install_4.1.2.sh –u USER
```

5. Hacemos efectivo el grupo de miembros:

```
$ newgrp sbox
```

6. Configuramos el entorno

```
$ echo 4096 > /proc/sys/vm/mmap_min_addr
$ echo 0 > /proc/sys/vm/vdso_enabled
```

7. Una vez instalado el scratchbox de maemo: instalamos el sdk

   http://repository.maemo.org/stable/diablo/maemo-sdk-install_4.1.2.sh

   ```
   $ sh maemo-sdk-install_4.1.2.sh
   ```

   Cuando empecemos a instalar el SDK aparecerán distintas pantallas, las opciones que debemos elegir son las siguientes.

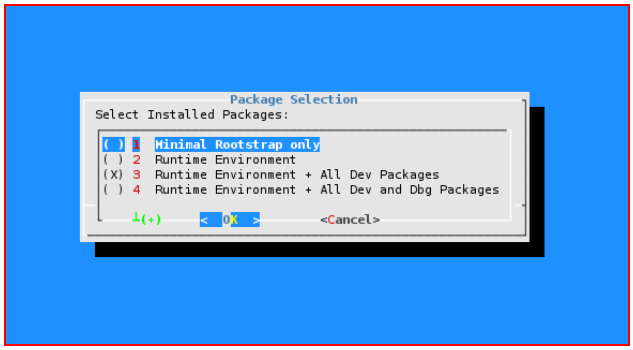


Ilustración 75 - Instalación SDK Maemo 1

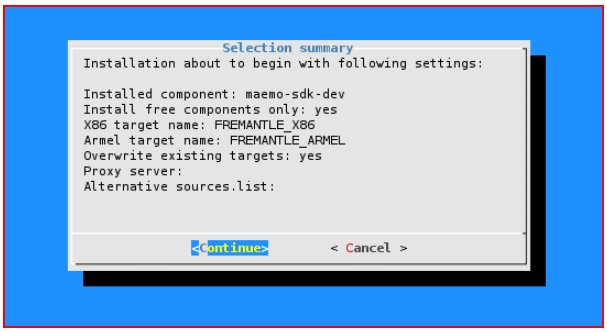


Ilustración 76 - Instalación SDK Maemo 2

8. Instalamos Xephyr

```
$ apt-get install xserver-xephyr
```

9. Una vez instalado Maemo procedemos a configurar el entorno. Para ello entramos en el scratchbox

```
$ scratchbox
      Welcome to Scratchbox, the cross-compilation toolkit!
      Use 'sb-menu' to change your compilation target.
      See /scratchbox/doc/ for documentation.
      [sbox-DISTRIBUCION_ARMEL: ~] >
```

10. Dentro de la consola de scratchbox instalamos los binarios de nokia:

```
[sbox-DISTRIBUCION_ARMEL: ~] > fakeroot apt-get install maemo-explicit
```

11. Establecemos el display:

```
[sbox-DISTRIBUCION_ARMEL: ~] > export DISPLAY=:2
```

12. Ejecutamos el Hildon y en la ventana del Xephyr aparecerá el Hildon:

```
[sbox-DISTRIBUCION_ARMEL: ~] > af-sb-init.sh start
```

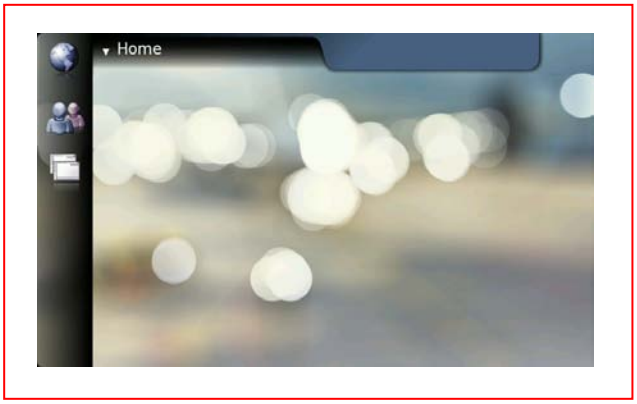


Ilustración 77 - Pantalla inicio Maemo

13. Añadimos la siguiente línea al /etc/apt/sources.list dentro del scratchbox:

```
deb http://repository.maemo.org/
fremantle/4bc37c7c77ebe90177c050b805a8dc79 nokia-binaries
```

14. Ejecutamos:

```
[sbox-DISTRIBUCION_<X86/ARMEL>: ~] > apt-get update
[sbox-DISTRIBUCION_<X86/ARMEL>: ~] > fakeroot apt-get install nokia-binaries
```

Nuestro entorno de desarrollo Maemo ya está listo para ser utilizado. Por ultimo, si deseamos desinstalar Maemo bastará con ejecutar los siguientes comandos:

```
$ /scratchbox/sbin/sbox_ctl stop
$ apt-get remove scratchbox-* --purge
$ rm -rf /scratchbox
```